**TCP/IP网络编程**

**课程综合设计报告**

**题目：屏幕共享**

**学院：计算机学院**

**专业：网络工程**

**班级：**

|  |  |  |  |
| --- | --- | --- | --- |
| **职务** | **姓名** | **学号** | **分工说明** |
| **组员1** | **何庆** | **202110420107** | **客户端** |
| **组员2** |  |  | **服务器端** |

**设计日期：2023.12.3**

**综合设计成绩考核表**

下表由考核教师按实际情况进行填写：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **考核内容** | | | **所占比例** | | **得分** | |
| 项目设计报告 | | | 50％ | |  | |
| 程序代码等的完成质量 | | | 30％ | |  | |
| 答辩及运行情况 | | | 20％ | |  | |
| **教师评语** |  | | | | | |
| **总评成绩** |  | | | 评定等级 |  | |
| **成绩与等级对应表** | 优  90－100 | 良  80－89 | | 中等  70－79 | 及格  60－69 | 不及格  59分以下 |

考核教师签字： 年 月 日

**目录**

# 1. 设计的任务及要求

## 1.1选题说明

该程序采用udp协议进行数据传输，使用TCP进行文件传输，实现主要功能为对客户端的窗口进行共享，其中用户可以根据自身的需求不同，共享不同的窗口。在客户端，将客户端的窗口进行展示。

基于UDP的屏幕共享涉及通过UDP协议实现实时屏幕图像传输，允许在网络上传输图像数据以便在远程端观看。这样的屏幕共享方案通常用于远程办公、在线协作、远程技术支持等场景。UDP协议相对于TCP协议来说，没有建立连接的过程，不保证可靠传输，但因此减少了一些协议开销，提供更低的传输延迟，适用于对实时性要求较高的场景。UDP是一种轻量级的协议，不保证可靠性，但在屏幕共享这样的实时传输场景下，一些丢失的帧可能并不会对用户体验产生过多影响，而带来的低延迟更为重要。屏幕共享基于UDP能够提供低延迟的图像传输，使得用户能够实时查看和操作共享屏幕。

Tcp相对与udp协议，会建立连接，保证可靠传输，会增大开销成本，传送的数据无差错、不丢失，不重复，按序到达。保证了传输的准确性，对于文件的传输层来说，需要确保不会有差错，采用滑动窗口协议来保证能及时接收收到的数据，进行流量控制。使用慢开始，拥塞避免、快重传、快恢复。在文件传输的场景下，使用TCP协议确保了可靠传输，不会产生丢包问题，差错问题。

## 1.2 程序涉及的知识点

图形用户界面：

图形用户界面是一种计算机用户界面，通过图形元素（如窗口、图标、按钮、菜单等）和鼠标操作，使用户能够与计算机进行交互。相对于传统的文本用户界面（cli），GUI提供了更直观、更友好的方式来操纵和控制计算机。该部分功能是使用图形和用户进行信息交互，避免了过多的冗余信息和复杂布局，使用户能够快速理解和使用，提高了用户的使用体验，以及交互体验。

多线程：

多线程是一种并发执行的编程模型，它允许一个应用程序中的多个线程并行执行，每个线程都拥有独立的执行流和局部状态，但共享相同的全局状态和资源。线程是操作系统调度的基本单位，一个进程可以包含多个线程，它们之间可以并发执行。

在多线程的环境中，每个线程都独立执行特定的任务，且线程之间可以共享相同的内存空间。这使得程序能够更有效地利用多核处理器的性能，并提高系统的响应性，特别是对于需要进行并行计算、处理异步任务或同时执行多个操作的应用场景。

该程序客户端包括两个线程，主线程与进行屏幕共享的子线程。主线程的功能主要包括实现与用户之间的交互，实现信号与槽的关联，以及创建子线程，同时，获取屏幕的截图必须在主线程执行，所以还需要向子线程传递数据。

子线程的功能主要包括，创建udp套接字，建立与服务器端的连接，发送数据，判断服务器端是否断开连接，以及与主线程之间进行通信。

定时器：

定时器主要用于处理周期性的事件，在该程序中的作用是重复获取屏幕的截图，向服务器端传递信息。

回调函数：

Qt的回调函数是一种特殊的函数，它可以把用户自定义的函数作为参数传递到类种，从而实现在类中调用外部定义的回调函数，实现类之间的调用，减少代码的耦合度，使代码的结构更清晰，使程序的灵活度和可扩展性更强。在改程序中，回调函数主要用于线程之间的通信，以及信号和槽的连接。

客户端算法：

客户端算法主要实现信号与槽的连接，当用户点击某个按钮时候，触发对应的回调函数，进而执行相应的操作。包括执行开始传输的“开始”按钮，“暂停”按钮，“文件传输”按钮，“停止共享”的按钮。每一个按钮对应各自的回调函数执行不同的功能。

服务端算法：

服务端算法主要包括，接收客户端传输的图像信息并显示在屏幕上，以及接收文件传输的按钮，以及文件保存路径。文件传输的进度条显示。

文件操作：

该程序中的文件操作主要设计以下内容：创建文件，打开文件，关闭文件，读文件，写文件。当文件数据从客户端传输到服务端，会为传输的文件创建一个新的文件，同时打开该文件，将数据写入到文件中，为文件重新命名，并关闭文件。该程序中，客户端以只读的方式打开一个指定的文件，并获取文件的名称，读取文件的内容。

# 2.需求分析

## 2.1功能要求

* 实时共享：用户能够即时共享屏幕上的内容，在接收端确保能够及时看到用户共享的屏幕信息。
* 多平台支持：提供跨平台支持，允许用户在不同操作系统，window，Linux，Mac设备进行屏幕共享。
* 权限控制：允许共享者选择性共享整个屏幕或特定应用程序窗口。默认情况下，会选择共享整个桌面窗口，用户可以根据自己的需求进行选择，不同的应用窗口。
* 协议选择：屏幕共享需快速传输数据，所以在协议上选择udp协议。Udp面向无连接，尽最大努力交付，没有拥塞控制，开销小，传输效率更高，更适合图像传输的场景。
* 高画质支持：提供高质量的屏幕共享，确保清晰度和流畅性，特别是在显示图像和视频内容。在带宽一定的情况下，画质越高，在传输时消耗的时间越多，用户需要考虑使用体验以及流量消耗。
* 文件传输：提供文件的传输功能，从而让用户可以向对方发送想要发送的文件数据，文件的发送由于需要考虑准确性，

## 2.2开发和运行环境，注意事项。

* 注意事项：性能优化，屏幕共享涉及大量图像数据的处理和传输，因此需要关注性能，使用合适的图像压缩算法、优化算法实现和网络传输效率。实时性，在屏幕共享中，实时性是一个关键问题，需要尽可能降低图像传输和显示的延迟。用户界面设计，需要设计一个直观、易用的用户界面，以提高用户体验。提供控制选项，选择共享区域，开始，暂停，结束等。考虑网络带宽控制，以防止过多的数据传输导致网络拥塞。
* 开发和运行环境：该程序的开发环境选择Windows11，开发框架选择qt,开发工具选择qtCreater，版本为5.12.12,编译器mingw73\_64,开发语言使用c++。主要使用Qt框架内置的网络模块，界面库选择qt Widets,提供易用的用户交互界面。qt框架兼容Linux，Windows，Macos，所以跨平台的问题有了很好的解决方案。

## 3.系统框图

客户端系统框

客户端系统框图：

# 4.系统协议设计

### 客户端

该项目中所进行屏幕共享使用的是udp协议，主要原因在于udp面向无连接的，能够快速传输大量数据，尽最大努力去交付报文，并且没有拥塞控制。对于屏幕共享需要在短时间传输大量数据的需求可以得到满足。在该程序中，为了实现udp套接字的通信，主要使用了QudpSocket类，QudpSocket类是 Qt 框架中用于实现 UDP 通信的类，它提供了发送和接收 UDP 数据报的功能。该程序主要基于该类进行封装，实现udp套接字的通信。

从客户端发送的数据，包含两种数据报，分别为图片的长度信息，和图片的数据流信息。在客户端，通过定时器定时截取到屏幕截图，将其传递给子线程，通过map函数，将其转换为jpg格式，同时去处理图片的清晰度，网络带宽一定时，传输图清晰度越高，所需要的时间越长，会造成卡顿。紧接着，计算出图片的长度信息，将信息通过封装，格式为“new imgage:xxx”，发送给服务端。在发送完成图片的长度信息后，程序会发包发送图片的数据信息。由于udp协议对于单次传输的数据不能过大，而图像信息长度远超过协议规定的长度，所以需要对图片进行分片处理，将图片的数据信息进行分组发送到服务端。分片的长度是一个预先设定的定值。

对于文件的发送，客户端会获取用户需要发送的文件的文件名，并计算出文件长度，同时获取需要发送的文件的文件名称。使用QDataStream将数据按照一定的顺序写入二进制流中， 顺序为“总大小信息空间，文件名大小信息空间，文件名”。再写入时，先需要写入文件名信息，在计算出总信息大小，写入数据流开头，之后通过write函数向指定的地址及端口发送数据。

### 服务端

服务端会对接收的数据进行判断，如果是接受的信息为图片的长度信息，则代表即将发送新的图片，此时会进行接收数据前的准备，包括初始化接收数据的数组，已接受数据的信息等。同时会将接收到的长度信息进行提取。

如果是图片数据信息，创建一个比特数组用于存放图片数据信息，之后通过memcpy函数去复制缓冲区的数据，同时进行判断当前接收的分包数据与总数居是否相等。当等待所有的分片发送完成时，在服务端进行组包，在将图片信息通过loadfromdata函数进行还原，最后将还原得到的图片在组件上呈现。

对于文件信息，服务端通过bytesAvailable函数判断接收的大小信息，满足条件则读取数据流中的文件大小信息与文件名信息。当数据接受完成后，当接受完以上信息后根据得到的文件名创建文件，同时拿到数据，将数据写入文件。如果文件已存在则覆盖。写入完毕后，文件保存在指定目录，关闭套接字。

# 客户程序设计

## 客户端界面

图片传输，用户点击开始（start）,会开启定时器，每经过一定时间间隔，截图屏幕。同时完成套接字的初始化，计算图片的数据信息，向服务端发送数据包。同时开始按钮变为暂定按钮，用户通过点击，可以实现中断数据报的传输，实现的主要原理为通过函数去中止定时器的使用。用户点击“仅共享当前窗口”,程序会自动去截取当前的页面进行传输，点击“共享其他窗口”,程序会根据选择的应用进行判断，进行指定窗口的传输。

文件传输时，用户点击文件传输按钮，会打开新的ui界面，包含文件打开和文件发送，点击打开按钮，打开系统文件选择窗口，选择完毕后，发送按钮变为可用状态。点击发送按钮，会将文件以比特流的形式进行传输。同时在右侧的进度条会实时更新当前传输的进度。

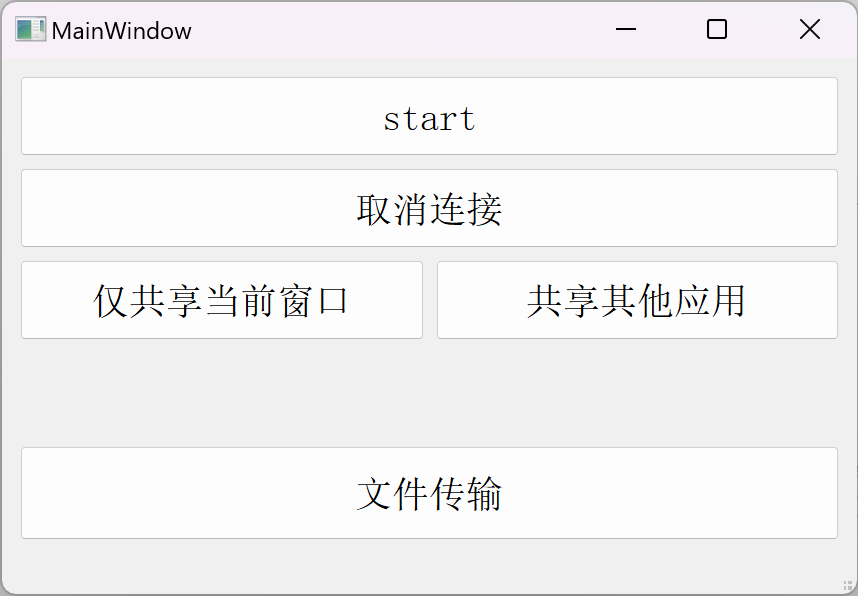


图5-1客户端ui

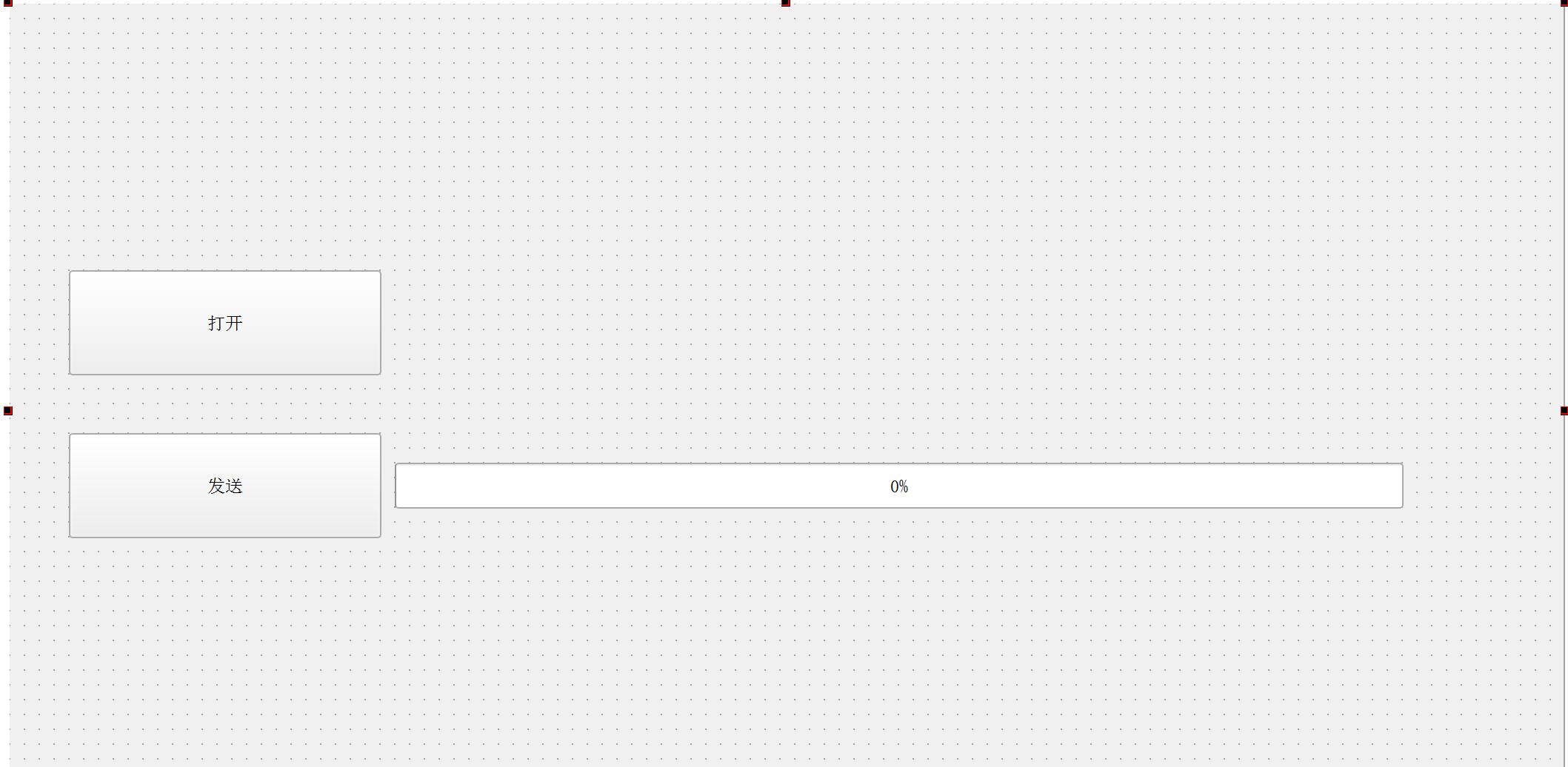


图5-2 文件传输ui

## 屏幕捕获模块

该模块主要包括屏幕获取的函数primaryScreen，通过该函数可以拿到原始的桌面指针，之后调用grabWindow函数，传入一个句柄。通过获取到桌面的句柄，拿到桌面的截图，接着使用信号与槽函数和，通过emit函数向子线程传递获取的数据。用户点击开始按钮，程序开始执行，通过定时器，间隔特定时间，调用截取屏幕函数getPic,用户点击暂停时，定时器中止，线程阻塞。

屏幕捕获流程：

## 数据传输模块

对于图像数据的传输，在子线程中，会创建一个udp套接字，获取网卡对象，加入udp多播组，设置最大跳数，以及缓冲区大小。之后通过互斥锁去管理线程的开始与阻塞。当接受到主线程传递的图片数据后，将其保存为jpg格式，计算数据的长度信息，将其传递到接收端。接着将图片数据分包发送到接收端。

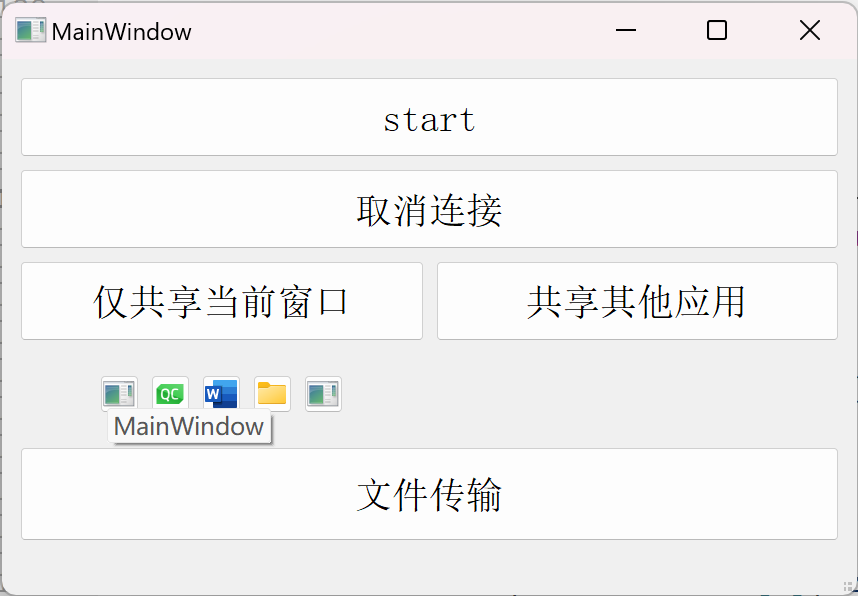
## 文件传输模块

首先，程序会创建一个tcp套接字，同时指定端口和地址，文件传输主要包括，打开文件选取窗口，程序会去读取用户选择的文件的内容信息。计算文件的长度大小，接着依次向二进制流中写入大小空间信息，文件名大小信息，文件大小信息等，之后向服务端发送数据。

## 共享指定窗口模块

在window系统下，句柄是每个窗口的唯一凭证，我们可以通过获取到窗口的句柄来拿到对应窗口的截图，共享其他的应用模块中，需要获取到状态栏的应用，主要通过函数EnumWindows。通过一个回调函数EnumWindowsProc枚举出屏幕上所有的顶层窗口。

通过拿到窗口的图标，标题，句柄。客户端的窗口显示出应用。与此同时为每一个图标设立一个槽函数，来处理点击事件的回调，当用户点击对应的图标，根据对应的句柄，调用截图处理函数，获取到指定应用的截图。同时开启一个定时器，重复获取应用的窗口截图。



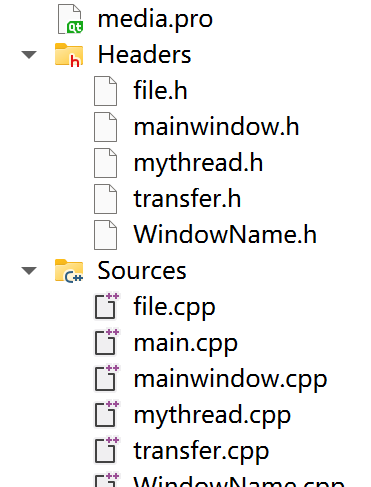
5.6-1 ui界面（共享指定窗口）

## 网络通信

客户端进行网络连接;首先，该程序预先设定了端口号以及组播地址，当程序进行udp通信时，会判断是否含有网卡，接着将添加组播组中，所有在当前组播的用户都可以获取到屏幕共享的信息。用户可以选择随时退出组播通信。同时用户共享时，会进行判断，是否有其他人正在进行共享。

## 工程代码设计

客户端目录结构：



File.h

主要声明槽函数，操作ui界面的指针，实例化传输模块的类。

Mianwindows.h:

声明截取屏幕相关的槽函数，定义信号函数，定义计时器指针，线程指针，存放句柄的变量，实例化文件类。

Mythread.h

定义udp套接字的指针，定义互斥锁，接收图片信息的变量。定义接收信号的函数，以及与线程操作相关的函数。

Transfer.h

定义文件ui的指针，创建tcp套接字的指针，定义文件相关的变量，包含文件总大小，文件名，已发送数据大小，每次发送数据大小，文件路径，数据缓冲区。以及定义相关槽函数。

File.cpp:

定义槽函数，分别为打开按钮与发送按钮，调用对应的函数。

Main.cpp:

程序运行的主体部分

Mainwindow.cpp:

程序运行的主线程部分，主要是定义屏幕截取的函数，按钮的槽函数的定义，以及任务栏应用的获取

Mythread.cpp:

程序运行的子线程部分，主要是定义udp套接字，实现将图片数据转换成比特数据发送。以及互斥锁的开锁与解锁。

Transfer.cpp：

程序文件传输部分实现。主要是将文件通过数据流发送，同时同步更新进度条的数据。

## 代码清单

1. \*\*项目 A\*\*

- 描述：这是项目 A 的简要描述。

- 示例代码：

```python

// 这是项目 A 的示例代码

def project\_a\_function():

# 实现代码

```

1.客户端

file.cpp：

#include "file.h"//引入头文件file.h

#include "ui\_file.h"引入头文件ui\_file.h

void file::**on\_openButton\_clicked**()//打开文件

void file::**on\_sendButton\_clicked**()//发送

mainwindows.cpp:

#include “mainwindow.h”//引入头文件

#include ”ui\_mainwindows.h”//引入头文件ui\_mainwindow.h

QVector<HWND> windowHandles; //储存句柄的数组

QVector<HICON> windowsIcon; //icon

MainWindow::**MainWindow**(QWidget\*parent):QMainWindow(*parent*),

ui(newUi::MainWindow) //构造函数

BOOL CALLBACK **EnumWindowsProc**(HWND hwnd, LPARAM lParam)

# 6.服务器程序设计

## 6.1程序结构：

客户端界面主要包含两个部分，分别为左侧的视频信息显示部分，与右侧文件数据接收部分。接收到传输的图像信息后，图像会在左侧显示，当有文件进行传输时，右侧进度条会显示进度，以及传输成功后，会进行确认接受。图片接受的功能实现主要包含,之后对数据进行识别，为图片长度数据时，对报文进行读取，获取图片长度并保存，若是传输的图片信息报文，则通过mcmcpy去复制缓冲区的数据，当接受的数据长度和图片长度等于后，将图片数据转换为图片。在组件上显示。对于文件数据的接收，首先需要设置二进制的版本，让通信双方保持一致，进行数据接受时，首先会先去接受总的数据大小数据，文件名称的大小，之后接受文件名，判断数据接收的长度，根据文件名去创建文件，以写入的方式打开文件，向里面写入数据，等待数据接受完毕后，关闭文件。

接收数据流程：

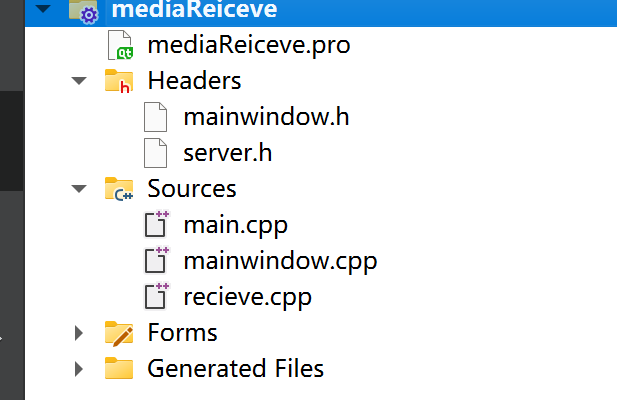


图6.1 客户端ui界面

## 网络通信：

创建udp套接字，将套接字绑定指定的IP地址以及端口，将套接字加入组播，通过readyRead信号判断是否有数据，之后调用相应的槽函数，服务器通过监听套接字接受来自客户端的连接请求对于数据的处理， readDtatagram去读取数据。

## 工程代码设计：



## 代码清单：

# 7.体会及总结

### 7.1 在设计及编程过程中遇到的问题及解决的方法。

**Udp丢包：**

Udp的丢包问题原因主要是由于接收对发送端的数据来不及处理，新的数据把旧得数据进行覆盖。解决方案是，将发送端的数据传速率进行调节，从而是得接收端能够对数据做出响应。

**线程数据传输的覆盖：**

由于更好的实现共享的效果，对屏幕的截取响应很短，将数据传输给子线程时，会出现将原有的数据进行覆盖，开启新的传输。解决方案是将子线程加入互斥锁，当数据开始传输时，将进程阻塞，从而保证进程不会因为新的数据到来中断。

**主线程和子线程的通信：**

由于对屏幕的截取只能在主线程运行，而程序数据发送在子线程中，需要主线程和子线程建立通信连接。解决方案是，使用connect函数，建立信号和槽的联系，主线程在需要通信的地方，使用eimt函数进行数据的传递

**子线程调用界面的ui：**

由于程序子线程通信过程中，界面会进行一定的提示，需要对主页面的ui指针进行调用。解决方案是，创建一个指针变量，在主线程的构造函数里面。通过将ui变量赋值给这个变量，从而在子线程中调用。

**多个ui界面如何实现：**

每个ui界面都要对应的指针，不同的ui指针控制着不同的ui界面，主要通过获取对应的指针，从而去调用对应显示和隐藏的函数，进而实现界面的开启关闭。

**获取状态栏应用句柄：**

句柄是窗口的唯一标识，获取句柄的方式主要通过EnumWindows函数去实现，该函数的回调函数会通过窗口的句柄去执行系列操作，可以获取到图标之类的信息。

### 7.2 收获及心得

通过此次项目设计，掌握了qt的使用，以及如何使用套接字进行通信，同时对于一个项目的设计有了大致的理解与掌握。在设计和实现屏幕共享程序的过程中，我深刻体会到了计算机网络和图形处理等领域的交叉应用。这个项目为我提供了丰富的经验，使我更加深入地理解了网络通信、多线程编程以及图像处理等方面的知识。我学到了如何使用套接字建立客户端和服务器之间的连接，以及如何在连接上进行数据传输。理解数据的传输过程对于确保实时性和稳定性至关重要。同时，处理异常情况的能力也是网络编程中的一项必备技能。为了支持同时多个客户端的连接，我使用了多线程编程。这使得程序更具并发性，但也引入了线程同步和资源共享的问题。通过这个项目，我学到了如何使用线程来处理并发操作，以及如何通过锁机制等手段避免潜在的竞态条件。在屏幕共享程序中，设计有效的协议用于客户端和服务器之间的通信至关重要。我学到了如何定义协议格式，确保数据的准确性和可靠性。

**文档常见的格式问题：**

**注：括号中的百分比表示存在这些问题的比例。**

**1、首行不是严格缩进2个字（95%）**

**2、序号不一致的问题。文档中同地位的序号采用了不同样式。（70%）**

**3、表缺乏表号、表名，且表号表名没有放在表的上面。（50%）**

**4、图缺乏图号、图名，且图号的编号很古怪。格式应该是“图x-y 图名”（60%）**

**5、文中存在大量空白，包括空行。（98%）**

**6、缺乏标点符号。（70%）**

**7、流程图当结构图。（50%）**

**要求：**

**1、除了章节标题，所有段落，包括序号，首行都缩进2个字。**

**2、不能有空白，如果是图导致空间不够，将图后面的文字提前填充空白。**

**3、因为章节序号是用了1，在序号中不能再用1这样的表示。**

**4、所有的图、表都必须给予一定的解释说明。**

**5、文中不能出现大量代码，必须用流程图和算法来说明。**