1. 尝试阶段：使用tcp实现了：局域网下、对话双方之一知晓对方的IP和port的文本对话
2. 尝试视频通话阶段：
   1. 尝试使用tcp实现视频通话。采用webcam不断while(true)进行拍照，每一张照片发送一个分组，进行视频通话。
   2. 发现tcp无论如何无法实现视频通话。
   3. 改用udp尝试实现视频通话。
      1. 开始时双方都开启serversocket等待accept。其中一方需要知道对方的ip和port，进行search操作。Search操作会向对方发起一个tcp连接。连接成功后，发起方的serversocket关闭，双方使用已经建立起来的tcp连接进行文本通信。
      2. 文本通信中，若任意一方发送“/video“，则双方各打开一个与当前tcp端口号相同的udp接收端，开启视频通信。
      3. 双方打开webcam不断拍照，每张照片作为一个udp分组发送到对方的upd接收端
      4. 此时尚未加上语音通信
      5. 双方此时的文本通信仍可使用，双方可以发送一些设定好的命名来终止视频通话或文本通话
   4. 在webcam拍照的同时，再开一个线程，接收麦克风信号，将每一小段麦克风信息包装成一个udp分组进行发送。给每个udp分组的开头再加一个判断字，用数字表示当前分组为audio或者video。Udp接收端在接收到udp分组时，先检查判断字，再分情况处理。
   5. 至此，支持text、video+audio的局域网IP电话基本完成。
   6. 此时，两个用户之间想要通信，需要满足：二者在同一局域网下、双方之一需要知道对方的IP地址以及port
3. 尝试不局限于同一局域网下的IP电话：
   1. 首先思考能否做成P2P形式。首先设想：对话双方开始时通过连接一个服务器来获取对方的IP地址和端口号，然后二者进行通信，不经过服务器。经过学习了解到：要使通信不经过服务器，需要复杂的手段完成nat穿透
   2. 决定一步步来，先实现经过服务器转发的IP电话。

**下面对“经服务器转发的IP电话“的设计和实现进行记录：**

对UDP：

1. server维持一个全线程可见的hashmap<user\_id,info-queue>其中info-queue是一个linkedlist<info>,代表将要发给用户的信息队列
2. 对每一个用户，维护一个线程，该线程的功能是：从该用户对应的queue中不断拿取数据并发给用户。
3. Server维护一个主线程，以及一个唯一的udp接收套接字。主线程的工作就是不断从udp接收套接字中提取分组，并检查头部信息，将其放入该去的info-queue中。

对TCP：

（想了想，类似QQ这样的通信软件似乎并不需要如此复杂？对于tcp文本信息，服务器只需要接收，然后检查其协议中规定的字段，然后再发给指定的对象即可，因为其不需要维护用户之间的一对一联络，client的图形化界面可以很好对收到的信息进行显示。客户端缓存数据，用户可以随时查看指定好友的信息。）

（以下的hashmap中的socket改为socketworker，socketworker中包含一个socket以及相应的objectinputstream和objectoutputstream。这是因为在将两个线程合并成一个线程的时候，需要同时获取两个socket的input和outputstream，如果只获取socket，再用socket生成，会发生报错。）

1. 服务器维护一个全线程可见的TCP Socket 的hashmap<user\_id,socket>。其key值使用数据库中的userId。
2. 每有一个用户登录，server用一个单独的线程进行管理,生成一个tcpsocket，再生成一个socket，并将其与用户ID一起放入hashmap。此时的键值对<user\_id,socket>对全部线程可见
3. 每个线程的while都会检查hashmap中是否还有socket，没有则终止线程
4. 当用户进入与另一个用户的对话中时，线程获取另一方的键值对<user\_id,socket>，将两个用户的键值对从hashmap中删除
5. 此时，server中，一个线程同时拥有两个用户的tcp socket。Server主持二者进行通信
6. 当用户发送特殊指令，打算开启audio时，server生成两个udp处理方法中提到的<user\_id,info-queue>，开启对应两个用户的udp thread。然后server使用tcp通知两边的client。
7. Client接收到特殊tcp分组，开始udp通信。

**至此为止，为了实现以上功能，需要规定以下应用层通信协议：**

**UDP：**

**1个int:代表目标用户的ID。**

**1个int:1代表video，2代表audio**

**TCP：**

**TcpMessage：int flag：**

**0：client→server，唤醒socket。**

**1：TcpLogin：client→server，进行登录。String username，String password**

**2：LoginResult ：server→client，回复登录。boolean isSuccess**

**（若登录成功）int userId，LinkedList<User> friends**

**3：CommunicationRequest：client→server，选择好友进行联系。int 好友ID**

**4：CommunicationRequestReflect：server→client，反馈选择好友的结果。boolean isFriendAvailable，int 该好友ID**

**5：CommunicationStartNotify：server→client，告知被发起者开启通话。int 发起者ID。**

**6：TextMessageFromClient：client→server，文本通信。int 目的用户的ID，String message**

**7：TextMessageFromServer：server→client，文本通信。int 源用户的ID ，String message**

**8：client→server，发起video。1个int：srcID，1个int：desID。**

**9：server→client，发起video**

**10：server→client，已准备好，可以开始video**

**11：client→server，终止video1个int：srcID，1个int：desID。**

**12：server→client，终止video**

**13：client→server，终止通信**

**14：server→client，终止通信**