个人总结报告

16337341 朱志儒

贡献：

在本次项目实践中，我实现了 c/s 通信并参与了 p2p 中 peer 端部分代码设计和 debug。

在 c/s 通信中，我设计了 client 和 server 之间的通信协议，确定它们通信的报文头，实现报文头的封装和解析过程。当然，我还实现了它们之间的数据传输，既可进行小文件的传输，又可进行 GB 大小文件的传输。该传输是稳定的，即使在传输超大文件的情况下也不会发生丢包的情况。

在 p2p 通信中，我参与设计 peer 中发送与接收分块文件函数的设计，我还参与 p2p 通信代码的测试和 debug 过程。在 debug 过程中给出了一些有意义的解决方案。

在项目中遇到的问题：

这个项目是在 TCP 套接字的基础上实现的。在课堂上，我们学习了 TCP 协议的特点，即它是面向连接的、提供可靠的数据传输服务、提供流量控制、提供拥塞控制、提供全双工通信和 TCP 是面向字节流的。当初，我以为 TCP 协议既然提供可靠的数据传输服务，那么我在服务器端发送多大的字节流，在客户端就应该立即接收相应的字节流。然而，在实践过程当中，我遇到了一个严重的问题，那就是当我从服务器端发送一个大于 8MB 的文件，客户端接收后，该文件发生缺失，即有大约 1%的数据丢失了。我测试过传输一个大约 27MB 的图片，客户端接收到的图片有 100KB 的数据丢失了，但还是可以正常打开图片，只是图片的底

部全是灰色。我也测试过传输一个 300MB 的视频文件，客户端接收后也有大约 10MB 的数据丢失了，但还是可以正常播放该视频，只是视频的最后几十秒不能正常播放，并且整个视频播放时没有声音。这就说明传输的文件越大，数据丢失率也会越高。这是为什么呢？刚开始我以为是 TCP 并没有提供可靠的数据传输服务。然而，我通过 wireshark 抓包后发现，TCP 确实将整个数据传输给客户端了，也就是说在传输层没有发生丢包，丢包是发生在应用层，这意味着我写的代码有漏洞。我就在客户端将每次接收的数据的大小打印出来，发现它并不总是等于 recv()函数的参数，也就是说它没有达到我设定的最大长度。这就意味着，服务器发送一段数据，有一部分数据到达客户端，而另一部分还在传输的链路上，然而 recv()函数是即时执行的，它不会等待在链路上的数据，TCP 将到达的数据立即上传到应用层，而我编写的程序在接收完这段数据后就会立刻关闭连接，这就导致有部分数据永远不会被应用层接收。找到这个问题的关键所在后，解决这个问题也就变得简单。我在有关接收数据的代码中实现已接收数据的总长度不等于文件总长就继续接收而不断开连接即可。