计算机网络期末大作业个人报告

颜泽鑫 15331348

分工情况

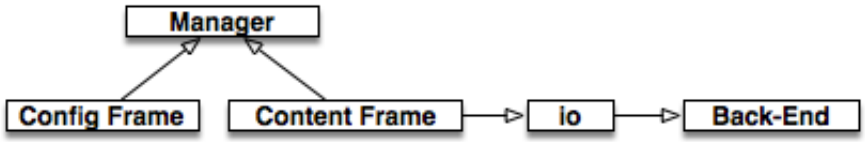
在本次项目中，我主要负责“前端”的部分，并且讨论项目具体模块分离。最后协助产品经理测试代码。

1. 讨论确定项目模块化。
2. 确定UI所使用的框架。
3. 设计UI界面，包括必要的功能（例如发送数据包、接受数据包、显示接受信息和日志信息等）以及有助于用户使用或debug使用的功能（例如显示路由表、邻居表、保存日志信息和保存接受数据包的信息等
4. 编写UI界面代码，包括后续补充拓展功能等。
5. 协助产品经理测试代码。

项目过程

1.

项目1.0版本时，我们还没有非常确定项目所需要的具体功能，因此在这个过程中，我一直在思考如何展示我们的项目。一开始我的想法是使用CLI（command line interface）。在我们这学期计算机网络实验课中所使用的路由器和交换机就是如此的。所有的命令都是通过命令行输入，结果也是通过命令行输出。但我很快地觉得这不是一个很好的选择。原因在于，如果使用CLI，那么就必然无法动态显示目前接受到的数据包信息，同样也无法动态显示debug信息（实验课中路由器的debug信息是在CLI中不断地显示，如果要停下来只能输入停止命令，但因为屏幕不断地显示debug信息，要输入停止命令就需要拼手速。）。我认为显示数据包信息和debug信息对于用户使用或则说对于后期测试而言是非常重要的功能。因此我们选择了编写GUI。我们所使用的GUI框架是wxpython。我从来没有写过桌面应用，因此这对我而言是需要现学现用的东西。不得不说，写GUI是非常恶心的事情，远远不如写Web优雅。但对于这个项目来说，为了界面再写一个后台就未免有点多此一举了。因此，我还是保持着耐心把这个GUI写下来了。庆幸的是，网上有十分丰富的示例代码，而且我们的需要比较基础，光是阅读和理解这些示例代码就足以完成这个项目了。我所涉及的UI界面框架如下，用manager来控制配置界面和主界面的跳转，用io来让算法和传输模块可以把debug信息输出到主界面中。



2.

整个UI界面的编写是比较顺利的（虽然代码很恶心）。但到了与后端代码整合时出现了一个很奇怪的bug。我们在io模块中保留了一个Content Frame的实例，当后端代码需要显示debug信息时就可以调用io模块中的函数，而这些函数会把对应的debug信息显示在Content Frame中。但奇怪的事情是，一旦运行整个程序时，程序在输出几条debug信息后就崩了，错误提示看起来与多线程有关。我和项目经历调试了一个多小时后，我们发现如果io模块中的函数如果没有向Content Frame中写入信息而是直接输出到命令行的话，就不会出现这个bug。而这也说明了bug的来源在前端。后来我们Google后发现，原来wxpython的Frame是不允许在非UI线程中修改UI的实例的。在后端调用io模块时所处的线程显然与UI线程不处于同一个线程，出bug也是显而易见的了。因此，解决的方法就是调用wxpython中内置的函数，使得后端对Content Frame的修改可以保留到UI线程执行时才进行修改。于是，fix bug！这又再一次说明，GUI的编写确实是非常不优雅的。在写Web的时候，如果要修改界面的输出信息时，哪需要顾虑这么多呢。

3.

最后，在测试过程中，我们发现增加显示路由表和邻接表的功能有利于发现bug和完成测试。因此，我们就在原有的基础上完成这部分的拓展。

个人总结

在本次项目我担任了“前端”工程师的职责。但其实我并不是一个热衷于前端的人，只是项目刚好需要有人完成这部分的工作，我才接锅的。在这个项目中，我没有过多地涉及算法部分的实现问题，而是专注于前端的人性化设计。在整个过程中，我们走了许多的弯路。例如在一开始，产品经理掉线导致负责后端的dalao们不知所措，盲目地写了很多难以debug和维护的代码，以致于我根本无从完成前后端的结合。后来产品经理上线了，细化了模块并且讨论清楚了各个模块的具体功能之后，项目才真正走上正轨。对于我而言，当然就是更容易完成前后端的结合了。这充分说明团队合作的重要性，UI界面是设计是需要讨论的，当然后端模块的分离也是需要讨论的，这绝不是一个人能过完成的。