计算机网络大作业个人报告

15331344 薛明淇

分工情况

在这次project中我做的很大一部分工作是——"产品经理"，听上去就非常像在队伍里划水的。不过我做的主要的工作是：

1. 明确任务主要目标（见报告）以及具体需求。
2. 确定程序实现语言与环境。
3. 设计程序每个模块的结构，确定每个模块的具体作用和大致的接口，分工实现。
4. 实现`Neighbors`，`NeighborTable`和`DataDispatcher`。
5. 主要的测试人员，并且帮助主要的程序员队友们定位bug。
6. 最终项目报告与幻灯片制作。

项目过程

在拿到这个作业的题目的时候的第一反应就是模拟一个动态的路由系统（不过好像后来看大家展示的时候感觉这个事情在大家看来又没有那么显然），没有像数据库大作业那样相对陌生的算法，所以整体感觉还行。

0

自告奋勇当了产品经理，理清一下整个作业的逻辑思路，跟队友们简单讨论了一下大致写了一份需求1.0版本，里面大致描述了程序的模块组成和之间的关系，然后让队友们来写，酱。不过几天之后发现这个“需求”还不够具体，导致队友们对程序有不同程度的误解，直接把所有逻辑上分离的事情揉在一堆写，导致之后写界面的队友工作很难进行，于是大家决定还是仔细讨论一下模块之间的结构和大家的具体分工。

1

之后花了大概几个小时，几个人一起brainstorm如何在大家可以掌控的范围内把事情的结构变得更清晰，那天的成果基本确定了所有模块的整体职责和结构，确定了各个模块的接口和调用关系，简单画了一份UML。其实我挺惊讶大家都或多或少有一些把代码耦合性降低的意识。

2

接下来明确了分工和接口职责之后剩下就是实现和测试的问题了，因为用的python又没有非常严格的工程规范，很多小错误再所难免。我在这一阶段的任务就大致是1. 把我自己的代码写完 2. clear一些仍然不太明确的情况 3. 对于遇到的新情况对于接口进行再次调整。虽然整个事情我相对来说比较清楚，不过具体写起来的时候总会遇到这样那样更加具体的问题，所以就需要进行再次调整。

3

在实现的阶段要求对model有一个简单的测试，之后再并入UI之后就准备设计拓扑图对整个事情进行人工测试。在纪律性写log的帮助下每次程序崩溃都能找到对应的人出来接锅，一起分析原因然后fix。这一阶段让整个程序的逻辑大概能够完成，不会出现任何崩溃之类的重大bug。

4

收尾，最后在算法上有个小插曲，因为我们有意在DV算法上没有使用RIP协议的“最大16跳”类似的抽象，发现处理宕机超时的情况意外的复杂，于是我们就开了另一个脑洞在小规模的拓扑下还算干净地解决了这个问题。然后就是设计测试场景，根据测试场景最后处理一些小bug，最后就是码报告做幻灯片等等的苦力活。

个人总结

关于这个作业我自己最主要的思考是产品经理的定位问题，我自己并不是专业的产品岗，在这次项目中的任务——可以看出来——也不是一个传统意义上的“产品经理”。其实自己之前做一个项目下来，在很多时候都会很希望会有一个人告诉我我具体要做什么要做到什么程度，甚至要写什么用什么写。如果整体的需求和清晰的模块设计结合起来，很多时候具体的实现就显得不是那么那么的重要了，就比如说这回虽然我们遇到的问题也很多，bug程序崩溃的情况也很多，不过其实没有一个瞬间会觉得“哇这个事情进行不下去了”，“哇垃圾代码把我坑到了”的感觉，绝大多数的错误都被限制在每个人自己的domain里，即使实现出了问题，或者要加入新的逻辑也相对来说会很容易。

之前跟老师聊了一下，老师说这回作业的本意其实就是要我们熟悉一下这些个路由协议和算法的工作，不过其实在我们这个分工下只有写算法的那一个同学达到了老师的目标哈哈。不过我觉得如果要做出一个能达成老师基础目标的东西，真正重要的事情就完全不是算法本身了，大到我如何在算法跑对的同时写更少的重复代码，我如何把算法这个变量从不变的系统中分割开，小到我如何尽量使数据包的分发更加流畅，逻辑更加清晰，再小到我如何正确处理很多并发访问的情况等等等等。当然这里虽然有“看低”算法的嫌疑，虽然我并不是这个意思，不过其实这个作业布置下来，总归不是让我们打三道算法题吧哈哈。

一定要说是跟计算机网络本身相关的事情的话，我觉得这次作业引导我去对计算机网络协议架构做一个更深入的思考。比如可靠数据传输和路由系统必定属于逻辑不同的两个层面，为什么IP的数据传输只能是"best-effort"而不能是"reliable"，在这个项目的过程中大致会有一些感觉了。当把网络层的路由逻辑抽象出来之后，各个其它的模块就会开始思考自己的可靠数据传输协议的实现了，例如邻居添加的过程中，就会思考通过什么样的协议可以让更新邻居状态的过程更为可靠，使主机之间的邻居状态尽量一致，而这个需求抽象出来的话其实也就是传输层（TCP）的逻辑了。关于协议设计，像如果把路由逻辑和传输逻辑混在一起的话就看不大出来，不过我们这里传输的部分跟其它功能模块完全分离的话，其实就有点现实世界中各种功能性协议（路由协议呀，消息控制协议什么的）构建在网络层协议之上，但又不完全在传输层中的影子了。

关于合作，这回我的队友居然没说：“搞那么多模块有什么屁用。”我还挺意外的，哈哈哈，以往在这种时候，弄出一些模块，搞他们的依赖关系什么的是会被很多身边的人鄙视的（不过就事后看来，写之前搞一搞这些玩意儿还是很有好处的）。另外还有一点小小的感悟，关于写干净的代码究竟有什么作用——其实就是让别人也能快速参与进自己的代码里来帮忙debug嘛——我看着队友们的代码如是想，不过这次虽然写的完整的代码相对少，不过因为要根据日志信息来定位bug，所以看的代码相对来说还是挺多的，比如说偶尔还要帮@写UI的dalao 排一排UI坑，偶尔要抓@写数据传输模块的dalao出来背锅什么的，至于@写算法的dalao 我也只能偶尔帮他想想算法了哈哈哈。

总的来说，这次的收获，相比于这门课以及这个作业本意要我们达到的，我会更关注一些工程实践上的问题吧，自己以前也不是特别喜欢搞需求搞报告什么的，这回感觉这些个事儿其实在“软件工程”里还是挺重要的。其次就是对程序的设计上可能有一些新的想法，像为什么python不是个容易上手的语言等等（完全不是开玩笑！）。再之后就偶尔像上面所说，思考思考历史的进程和前人（计算机科学家）的工作，最后可能就是在这些事情上锻炼锻炼自己的耐心磨练磨练自己的脾气吧。这也还是个挺有趣的经历呢。