一年的工作回顾

2022-08-27

来到公司一年多了，想要简单做一点阶段性的回顾，因为平时会把大致的工作内容记录在内部的 Confluence 上，所以总结起来会有据可循。

一年的时间内发生了很多事情，我交到了一个很好的朋友，也因此在工作上不是那么专心，相比以前不管是工作效率还是用心程度都大打折扣。不过总的来看仍然有很大收获。

State channels

来到公司的第一个月主要是熟悉项目，这个项目比之前公司的项目规模大很多很多，功能上相当于整合了 Filecoin、IPFS、Raiden Network、Ontology 等公链项目，还自己实现了类似于 libp2p 的网络模块和网络代理。一开始看起来还是有点吃力的，一方面因为代码体量大，另一方面因为确实不了解公链，虽然知道区块链本身的技术模块，但不知道什么是 Layer 2、IPFS、PoC。第一个月把项目搭建运行起来，了解了上传下载的基本流程，主要看了 p2p 网络在协议层面的交互实现，第一次知道了 DHT 是什么意思。

第二个月除了深入熟悉代码细节外，做了一件事情就是把 State channels 中的路由查找从 DFS 换成了 Dijkstra 算法。项目里有一个类似于 Raiden Network 的 Layer 2，用来解决文件下载过程中要对其他节点频繁支付的问题。改路由查找是因为这个部分相对独立，不会给整个项目带来麻烦，至于 Dijkstra 算法可以考虑路径长度做出选择而 DFS 不能，这点优化其实没有意义，因为我们 DNS 节点不会那么多，这个和 Raiden Network 完全 P2P 的模式是不一样的。

后来两个月也就是第三、四个月，主要做的事情是在 State channels 中增加手续费，在中转节点上扣掉一部分转账金额。那是一个痛苦的过程，因为我当时不懂 Layer 2 也不懂 State channels。好在用了两个月的时间还是把协议搞明白了，由于项目场景的限制不能由发起转账的节点直接验证交易，引起一些 channel 状态不稳定的情况，不过无伤大雅。

进入公司后我几乎没有请教同事关于项目的情况，不管是项目的整体架构还是具体的代码细节，全部是自己去看代码、查文档，尤其是要面对很多自己完全不了解的概念，这样做当然是有意而为之。全部自己折腾效率低是理所当然的事情，不过好处就是，在那个过程之后我可以有足够的信心，仅凭自己的实力就可以搞明白那样规模的项目，完成该做的事情。

因为我是第一次跳槽，我对自己的能力是疑惑的，不知道在之前公司的感觉是错觉还是事实，我觉得之前公司的项目不行、技术也不行，而且跳槽是有工资的增长，我多少有点心虚，想知道在进入一家新的公司后，我能不能够称职地独立应付起这样的项目，有没有实力对得起工资。请教同事是多么简单的事情！在之前的公司我也受到了很多关照，不过我希望自己有独当一面的能力，这正好是个机会，必须不依赖外部帮助去解决问题了。

Solidity 合约

第五个月，我在纠结 Layer 2 该往什么方向优化，有点难以下手，正准备解决性能低下的问题，但也没有思路。后来得到一个需求是把项目里的原生合约用 Solidity 写一遍，因为后续有想支持 EVM 的计划。当时我不了解以太坊也不了解 Solidity，花了大概一周时间看 Solidity 文档。

第六个月也就是今年一月份，用 Solidity 重写合约，那其实是一段愉快的时间，因为不太需要思考做什么、怎么做，照着现成的写就行，产出的代码量还大。只是用不同的编程语言，你知道的，换编程语言没什么压力。

第七个月今年二月份，在节点的 SDK 中加入对以太坊 SDK 的支持。折腾了一下以太坊的测试网发现不太好用，最后还是先用节点的开发模式了。

Solidity 虽然语法容易理解，但是由于 EVM 的限制，也有很多需要注意的问题，以及很多语言上的细节需要时间不断熟悉，当时用一个月写完合约，后面却断断续续用了不少时间去修改完善。总之在那个过程里，那个需求上，我学会了写 Solidity 合约，熟悉了以太坊智能合约的开

发，尽管对于生产级别的合约安全问题还缺少经验。

文件夹上传下载

今年三月份开始，做的一件事情是文件夹的上传下载。这个想法的起点是项目对 Git 的支持，想要支持 Git 协议，能够直接用 git 命令克隆 Git 仓库，结果发现普通文件夹的上传下载都没有，只有对文件的上传下载。

我们的项目用了一部分 IPFS 的 IPLD 协议，把文件转换成块进行传输，但是没有用 IPFS 的文件管理部分。IPFS 有一层针对文件系统操作的 API，可以统一处理文件和文件夹，数据结构之间还能方便地互相转化。我们是直接读取文件转化成块的，这也给文件夹上传下载的实现增加了难度。

这件事情一直断断续续持续至今，因为总是不断有各种各样的小问题出现，也感谢测试同事耐心的配合。文件夹的处理比单个文件复杂一点点，因为文件夹会存在无限的嵌套，文件夹内同时包含文件和文件夹，子文件夹内还会有文件和文件夹。以及其他问题像空文件夹、大文件的处理。

IPLD 节点储存在 Merkle DAG 的数据结构中，单个文件会生成一个 Merkle Tree，而文件夹需要做的是在上传的时候，把多个 Merkle Tree 组织起来成为同一个树，然后在下载的时候根据这个树反序列化成文件夹、把内容写入到磁盘上。

其实实现思路是简单的，这个树结构中有数据块 raw node 和用来做中间节点连接数据块的 proto node，只要把文件夹相关的额外信息写到 proto node 的 links 中，就可以把文件之间的关联信息储存传输到其他节点了。

不过在具体的实现过程中花了不少功夫，也绕了一些弯路，比如上传生成块的时候因为添加了额外的数据导致块数据验证不成功，不能生成完整的树结构，不得不深入到 IPLD 的代码里 debug；下载的时候对块数据的解析不熟悉，也没有意识到节点之间块数据的传输是没有顺序的，在功能不成功的时候一度怀疑整体思路出了问题。

由于块数据的传输是无序的，就需要在下载的时候自己整理块的顺序，排序过程中因为搞混了树的层序遍历和前序遍历，一开始深度优先生成顺序发现总是有几个块的位置错乱，debug 了好久才定位到问题改成广度优先。

这是一个无关紧要但是有意思的功能，在这个过程中加深了对文件的上传下载的了解。

文件的非对称加密

今年四五月份疫情严重，居家办公一个多月，做的事情是文件对非对称加密的支持，之前只支持 AES 的对称加密。

这个没太多可说的，就是要注意 ECDSA 是数字签名算法，没有加解密这回事，要用 ECIES 之类的混合模式，结合对称加密去实现对文件的非对称加解密。

支持以太坊账户

今年六七八月份，重点关注新 Layer 2 的方案，想要实现基于 Optimistic rollups 的 Layer 2。

Rollups 部分还没怎么动，目前停留在存储节点对以太坊账户的支持上。这一段不短的时间内，除了编译运行一下 Optimism 的项目、增加储存节点对多种网络模式的选项，还花了不少时间完善之前的 Solidity 合约、解决像合约大小超过限制需要拆分、节点真实运行过程中合约结果和预期不符等问题。

由于账户地址和公私钥都变成了另外一种格式，储存节点的协议消息也需要使用另外的签名方法，这些内容的改动都还在进行中。

总结

其实没多少事情，但也没怎么闲着。不算太认真，但也收获很多。