NApp Store:
一个去中心化的应用平台

Nebulas

2018 年 12 月
版本号:1.0

目录

1 概要

从 2008 年诞生，比特币逐渐受到越来越多的人的认可，这也证明了 “去中心化” 这一想法的可行性。“去中心化”，很大程度上解决了现代社会所需的信任问题，通过技术提供了廉价的构建信任的方式。

以太坊更进一步的把 “去中心化” 这一思想从数字货币领域扩展到了任意协议，引入了图灵完备的智能合约。截止目前，智能合约不管在数量还是在多样性方面都证明了其巨大的潜力。更具体来说，去中心化应用有以下两个特性，1，应用行为的透明性，即，用户能够看到并理解去中心化应用的行为，去中心化应用不能通过宣传等手段夸大、捏造或歪曲应用的行为；2，应用相关的数据的透明性，即，用户能够理解去中心化应用所使用的数据，并不会造成额外的信息泄漏，这对于用户的隐私保护而言是十分重要的。

然而，也应该看到，目前的智能合约依然存在诸多问题，例如，用户使用智能合约必须配合相应的钱包或插件，智能合约提供的功能依旧十分简单等。究其原因，我们认为智能合约很难应对用户日益增加的对 “去中心化” 应用的的需求，这是因为：

- **智能合约所涉及的操作相对简单** — 这是因为智能合约的所有结果都需要上链，而目前区块链提供的 TPS 还十分有限，因此，难以在智能合约中进行相对复杂的操作；
- **智能合约的交互相对简单** — 目前，用户与智能合约的交互需要通过 RPC 的方式，这极大的限制了更加复杂多样的交互需求，例如，用户难以进行图形化的操作。

为此，我们提出 NApp Store（Nebulas decentralized Application Store），试图提供一个更好的去中心化应用平台，这包括适用于用户的客户端，NApp Store Client，也包括用户开发者的开发套装，NApp dev tools。

注意，我们并不使用 DApp 这个术语，而是使用 NApp。从概念上来说，二者皆指去中心化应用，但是，DApp 一定程度上被广泛的认为是完成一定功能的智能合约及其上的前端，而本文描述的 NApp 则不限于通过智能合约或前端实现。因此，本文使用 NApp，以做区分。

本文首先描述了去中心应用平台 NApp Store 的整体架构，然后，更进一步的介绍了客户端、开发者工具及一些相关问题的讨论。

2 整体架构

NApp Store 整体上分为三个部分，Nebulas 公链、NApp Store Client，NApp dev tools。Nebulas 公链做为基础设施，已经开发完成，不在本文的描述范围内。NApp Store Client 做为客户端，安装在每个用户的计算设备上，由用户直接使用。NApp dev tools 仅由开发者使用，是开发 NApp 所必要的生产工具。

概括来说，开发者使用 NApp dev tools 开发具有一定功能的 NApp，并将生成的 NApp 提交到 Nebulas 公链之上。NApp Store Client 通过同步最新的区块，得到开发者开发的 NApp，并将该 NApp 对应的信息展示给用户。最后，用户在 NApp Store Client 中选择想要执行的 NApp，并执行相应的 NApp。

开发者提交的 NApp 以 LLVM IR 的形式上链。开发者可以使用任意的、能编译生成 LLRM IR 的语言开发相应的 NApp，通过 NApp dev tools 编译生成相应的 LLVM IR，并最终通过 NApp dev tools 将 LLVM IR 提交上链。

NApp Store Client 接入 Nebulas 公链的 P2P 网络，实时同步 Nebulas 公链中的区块，进行必要的区块信息验证，提取区块中的 IR，并通过 UI 展示给用户。在用户通过 NApp 的相关信息，选择想要执行的 NApp 之后，NApp Store Client 将用户选中的 IR 与必要的库链接生成本地化的可执行应用，并执行。

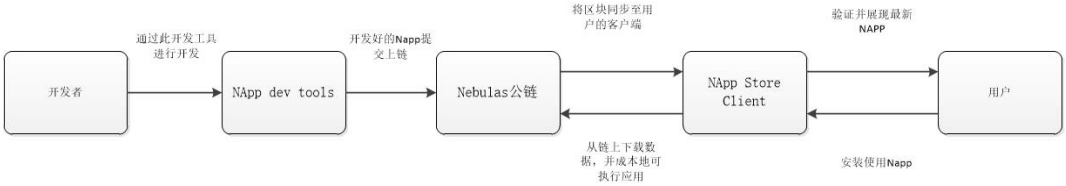

图 1: NApp Store 流程图

2.1 NApp 在链上的数据格式

此处，我们给出 NApp 在链上的数据格式，以表明开发者需要提交哪些信息，并为后续的 NApp Store Client 的展示提供必要的数据。

我们使用 protobuf 的格式描述链上的数据结构。

```
message napp_dep_meta{
    string name = 1;
    uint64 version = 2;
};
```

```
message napp_data{
    string napp_name = 1;
    uint64 napp_version = 2;
    string napp_intro = 3;
    string napp_icon_url = 4;
    repeated napp_dep_meta napp_depends = 5;
    bytes napp_ir = 6;
};
```

其中，napp_name，napp_version，napp_intro，napp_icon_url 的含义是明显的，需要注意的是 napp_name 及 napp_intro 皆有一定的长度限制。napp_depends 定义了该 NApp 所依赖的库及版本号，napp_ir 为该 NApp 对应的 LLVM IR。

另外，需要注意的是，由于一个 transaction 附带的 binary data 的长度通常有大小限制，因此，napp_data 的总长度不能超过对应的上限。目前，Nebulas 主网对这一限制为 128KB，这一限制可以进一步放松或解除。

2.2 NApp 能做什么?

NApp 并不是简单的智能合约或者简单的与智能合约的交互，NApp 的能力很大程度上由其所依赖的库决定。最基础的部分，是一个 NApp 具有本地的与 NAS 账户、Nebulas 主网及 Nebulas 主网之上的智能合约交互的能力；更进一步的，NApp 能够使用图形界面与用户进行交互、能够使用引擎创造更加本地化的应用体验。

举个例子，用户可以在 NApp Store Client 内安装一个书籍 NApp，并在书籍 NApp 内完成支付、阅读、甚至评价等操作。需要注意的是，其中的支付环节使用了书籍 NApp 与 Nebulas 交互的能力，阅读则使用了本地化的图形界面，最后的评价操作甚至涉及中心化的社交网络交互操作。从这个例子中不难看出，NApp 能够在使用区块链方面创造出流畅的用户体验。这将完全不同于现有的 DApp 需要使用额外的钱包或钱包插件的用户体验。

另外，从开发者的角度来看，开发 NApp 仅需要考虑本地化的开发及用户体验。这也有别于传统的 DApp 的开发，在 DApp 的开发中，由于智能合约不能提供本地化的交互体验，因此，开发者需要通过前端（移动客户端或网页客户端）与智能合约进行交互，更复杂的，前端需要通过开发者自己架设的中心化服务器与区块链进行交

互，这很大程度上增加了开发者的负担，同时也影响了用户体验。

2.3 NApp 与 DApp 的区别

从概念上来说，NApp 与 DApp 都包含了区块链交互的部分及用户交互的部分，其中区块链交互的部分通过智能合约，而在其他部分，这两者并不尽相同。

对于一个 DApp，一般通过 Web，或本地的客户端向用户提供交互接口，而 Web 或相应的客户端通过外部的钱包软件或插件与区块链进行交互，在一些情况下，Web 或相应的客户端也可以通过中心化的服务器与区块链进行交互。此时，由于缺乏统一的与区块链交互的方式，使得 DApp 在使用过程中，需要依赖外部的钱包软件或插件；更进一步地，Web、不同客户端之间的差异也使得区块链的使用难以有一致的开发或使用体验。

NApp 则基于 NApp dev tools 包含的 SDK 与区块链进行交互，并使用其中的 SDK 构建图形化的用户交互界面，而进一步的流程及细节则由 SDK 提供保证，既保证了开发的简易性，又提供了体验的一致性。

另外，相对于 DApp，NApp 的一个重要能力是能够以去中心化的方式完成自动升级。对于使用了 Web 的 DApp，可以通过在服务器中部署新的代码，而对于使用了客户端的用户，则依赖于用户的主动升级。

3 NApp Store Client

NApp Store Client 是用户使用 NApp 的入口，安装在用户的终端设备[1]上，包括以下组件

- P2P **网络** — P2P 网络负责接入整个 Nebulas 主网的网络，同步 Nebulas 主网中的区块；
- **区块验证** — 负责对接收到的区块进行必要的验证，丢掉不必要的、无效的区块；
- NApp **管理** — 从经过验证的区块中读取 `napp_data`，并存储在本地的数据库中，建立相关的索引信息；
- **资产及私钥管理** — 这不是 NApp Store Client 的主要功能，此功能仅仅是为了用户在使用 NApp 的过程中，能够进行必要的支付操作；

[1]如用户的 Mac，Windows 及 Linux 设备，甚至 iOS 及 Android 设备，此外，我们应该优先考虑 Mac 设备

- NApp **依赖库** — 包含一个 NApp 运行所必要的程序库，如 UI、发起交易等；
- UI **交互** — NApp Store Client 将包含一个完整的、图形化的用户交互界面。

NApp Store Client 从功能上来说包括三个部分，NApp 的管理、NApp 的安装及运行、资产及私钥管理。

NApp **的管理** NApp Store Client 可以认为是 Nebulas 主网的一个轻节点，即，会同步所有的区块信息并进行必要的区块信息验证，但是不会成为出块或验证节点，亦不具备任何的挖矿功能。这一方面是因为用户的终端机性能较弱（包括硬件和网络），另一方面，也因为在终端机上进行开销较大的出块或验证操作会影响用户体验。

NApp Store Client 仅仅保存必要的 NApp 的信息，而不会保存所有的区块信息。在同步、验证完所有的区块信息后，NApp Store Client 会提取其中的 NApp 信息，而抛弃区块内的其他信息。这主要是因为区块数据过为庞大，在用户终端中保存如此规模的数据会极大的影响用户体验。

NApp Store Client 会保存所有的 NApp 信息，包括同一个 NApp 的不同版本。这是为了让用户能够选择不同的版本。

NApp **的安装及运行** NApp Store Client 会在 UI 中展示所有的 NApp 及 NApp 相关的信息，如 NApp 的图标、名称、描述及版本号。用户可以有选择安装指定的 NApp。

NApp 的安装过程实际上是使用 LLVM 相关工具将 NApp 相应的 LLVM IR 编译为本地代码，并与 NApp 依赖的库链接成本地的可执行文件的过程。相应的，NApp 的运行实际上是用户直接运行生成的、本地的可执行文件。

资产及私钥管理 为了提供本地化的支付体验，NApp Store Client 需要管理一定的资产及相应的私钥。需要注意的是，NApp Store Client 本身没有更多的资产或私钥管理的需求，更多的功能是仅仅是可选的，而不是必须的。

4 NApp Dev Tools

NApp dev tools 供开发者使用，包括但不限于

- NApp SDK
- **资产及公私钥管理**

- NApp **开发及调试工具**
- NApp **提交工具**

5 几个问题

NApp Store 需要面临很多问题，此处仅讨论一些必要的权衡以及可能产生的影响。注意，以下所做的权衡都是根据目前的现状得出的，随着 NApp Store 的发展，这些权衡可能需要重新考虑。

5.1 关于付费 NApp

目前，我们认为所有的 NApp 本身都是免费的，这很大程度上是因为使用付费的手段购买 NApp 较为复杂，而实现这一复杂的功能，即，实现付费 NApp，并不能带来可预期的收益；另一方面，开发者的收益可以通过 NApp 内的支付或者开发者激励协议（DIP）来保证。

5.2 支付流程

我们认为，每个 NApp 都需要一定的支付场景，因此，我们希望每个 NApp 的支付体验是一致的。为此，需要 NApp Store Client 提供统一的支付流程，以库的形式提供给 NApp 使用。

5.3 NApp 升级流程

一个 NApp 实质上是一个具有一定标识的 LLVM IR，且该 LLVM IR 是放在链上的，因此 NApp 的升级是通过 NBRE 的能力完成的。

6 总结

本文给出了一个去中心化的应用平台，NApp Store，以及相关的 NApp Store Client、NApp dev tools 及一些相关问题。NApp Store 不仅可以做为 Nebulas 之上的应用平台，亦可能成为其他公链、甚至联盟链之上的应用平台，充满了可以探索、合作并进行拓展的空间。