软件架构文档

Cealgull

软件架构文档

版本 <1.0>

修订历史记录

日期	版本	说明	作者
28.06.2023	1.0	初始版本	吴逸洋，徐轲

软件架构文档

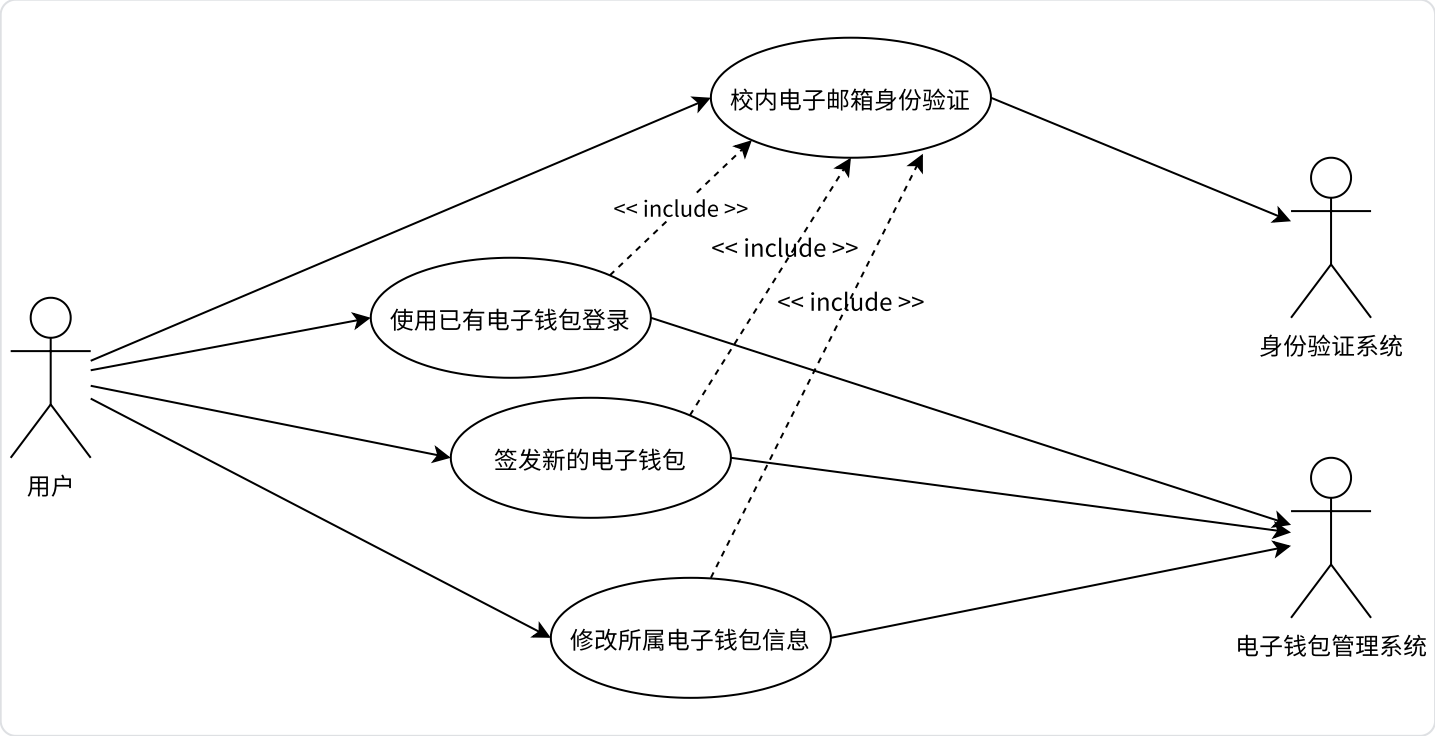
1. 简介

目的

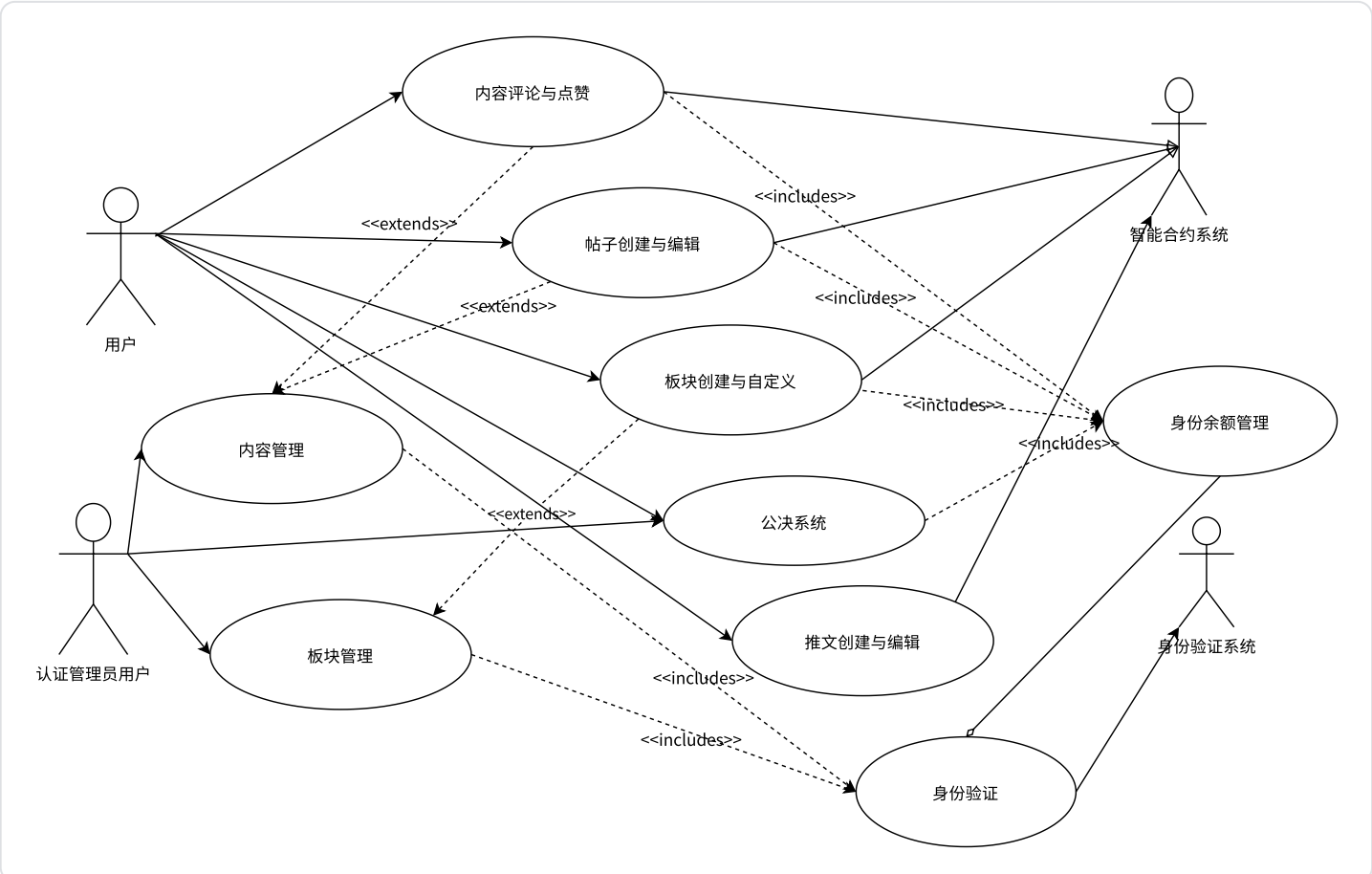
本文档将从构架方面对Cealgull系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

2. 用例视图

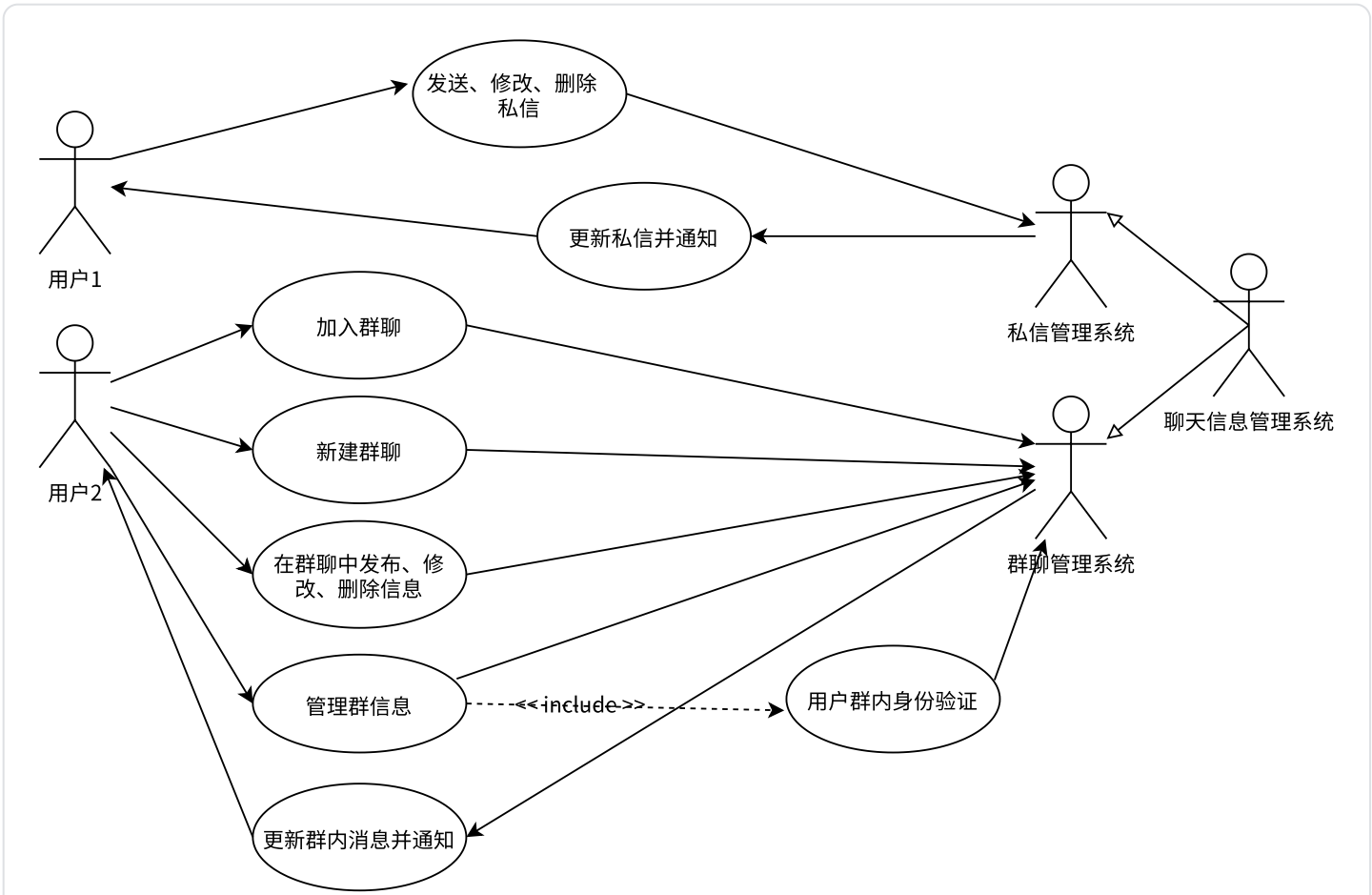
2.1 身份系统



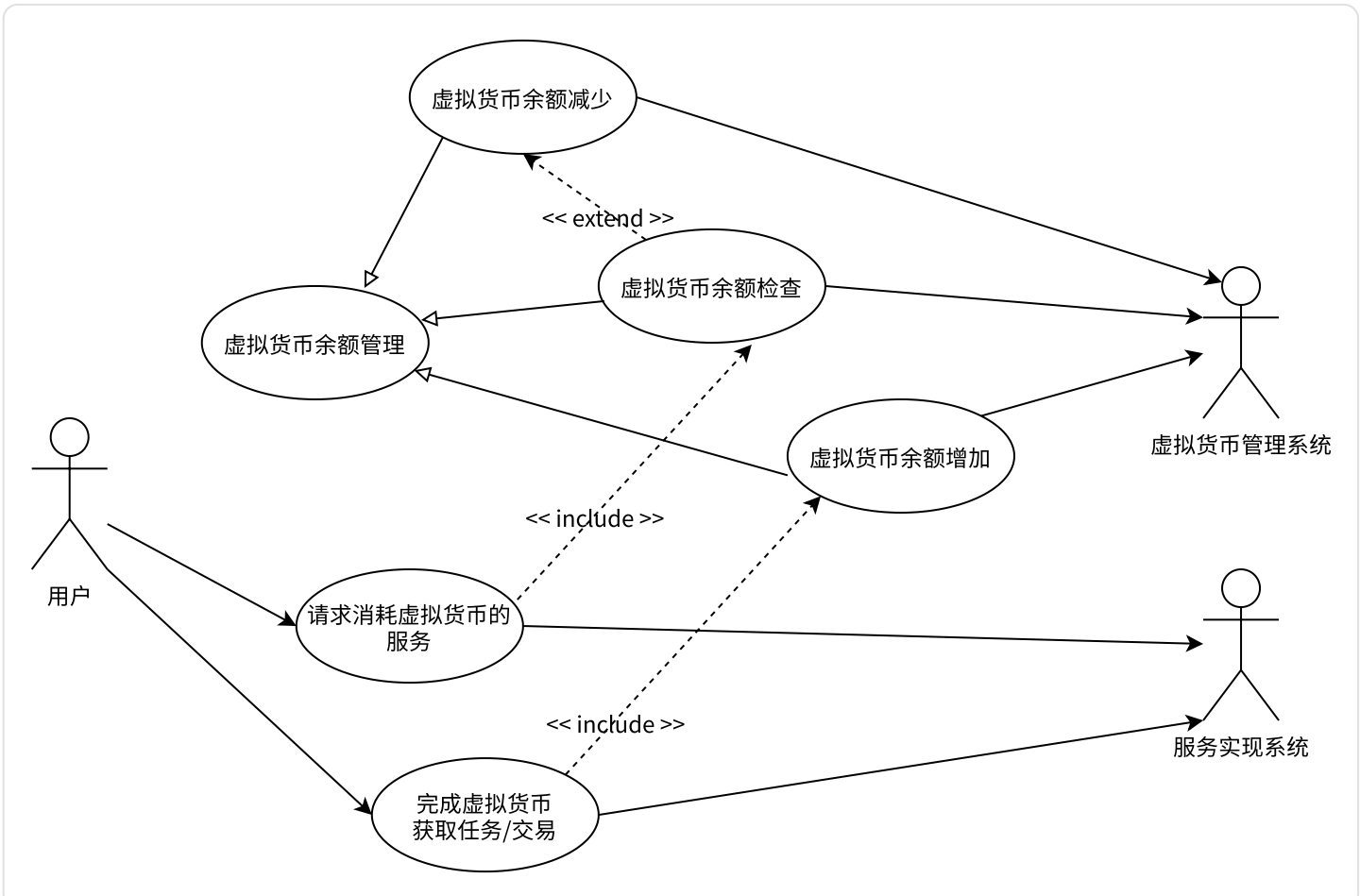
2.2 社区系统



2.3 私信以及群聊系统

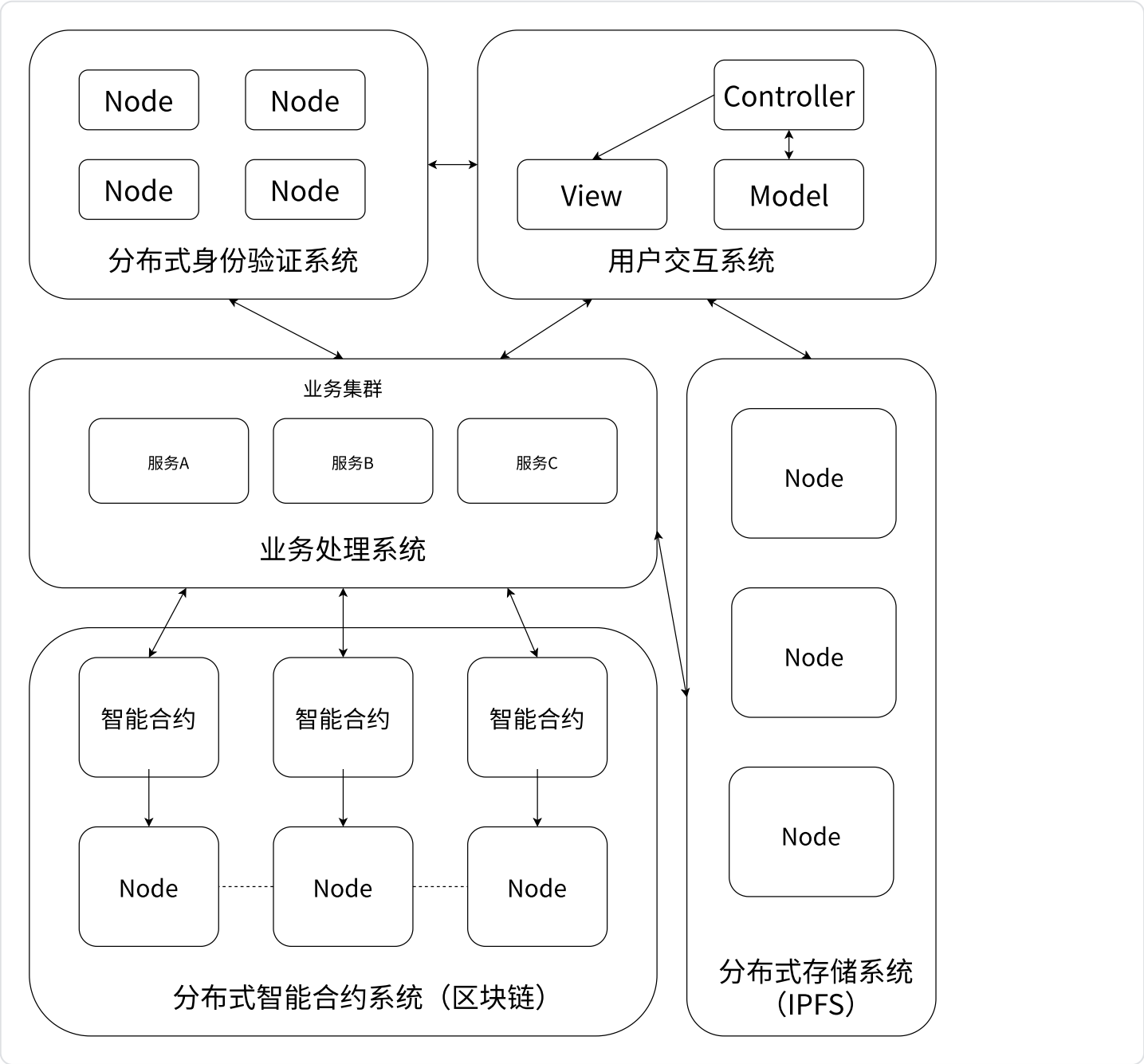

发送、修改、删除
私信
更新私信并通知
用户1
私信管理系统
加入群聊
新建群聊
在群聊中发布、修
改、删除信息
管理群信息
<< include >>
用户群内身份验证
更新群内消息并通知
用户2
群聊管理系统
聊天信息管理系统

2.4 货币交易系统


虚拟货币余额减少
<< extend >>
虚拟货币余额检查
虚拟货币余额管理
虚拟货币余额增加
虚拟货币管理系统
用户
<< include >>
请求消耗虚拟货币的
服务
<< include >>
完成虚拟货币
获取任务/交易
服务实现系统

3. 逻辑视图

我们所提供的服务包含以下几个子系统：



- 分布式身份验证系统
- 分布式存储系统
- 分布式智能合约系统
- 业务处理系统
- 用户交互系统
- 洋葱路由系统（可选）

概述

3.1 身份验证系统

我们使用邮箱验证 + 证书签发的方式来管理用户信息，当用户使用校内邮箱验证身份后，分布式的CA将向用户递交签发过得CA证书，用户后续将使用该CA证书来进行对于分布式智能合约的交互。分布式CA包含每个组织（学校）所维护的根CA和中间人CA，CA系统来指派中间人CA来进行身份服务。同时任何人也可以通过向邮箱验证的方式向根CA发出验证申请并自行配置CA系统并并加入CA系统，CA系统相互并不知晓身份，以做到去中心化的身份认证。

3.2 分布式存储系统

我们使用IPFS-Cluster来搭建私有的IPFS的分布式存储网络，包括在中间人CA间同步证书，在各个peer节点之间同步帖子、评论以及多媒体内容。我们可以通过设置IPFS-Cluster的Secret Key来进行管理，同时在同个channel的合约上部署一个私有的IPFS-Cluster来提供存储服务，这些数据分布地存储在peer节点上，并不对外界揭晓。同时在ipfs上，我们也可以部署相应的数据库系统来进行数据的组织以及处理。

3.3 分布式智能合约系统

我们使用Hyperledger-fabric搭建虚拟的货币交易系统以及其他的业务数据交互的逻辑的fingerprint，数据的详细信息都记录在channel上部署的ipfs集群，同时将智能合约的服务打包成外部服务经由peer分离，我们就能做到在小规模的对智能合约进行分发。当经由验证的外围用户想要加入peer节点进行服务时，他们便可以直接连接到智能合约服务器或者通过libp2p的pub/sub机制来订阅智能合约包。当上游开发者想要部署新服务时，智能合约服务器可以直接订阅到智能合约的新版本并直接上线使用。每一个fabric的peer都包含一个Membership Security Provider来限制用户触发智能合约的权限以达到智能合约的权限控制以及用户身份管理。

3.4 业务交互系统

我们使用Hyperledger-firefly来搭建对私有的区块链以及公有的区块链进行访问，通过其提供的命名空间来分离API访问权限，并使用内部所提供的MSP来对用户的信息进行鉴权来确定REST的执行权限，同时Hyperledger firefly同时支持接入websocket接口来做到区块链的时间监听。最后firefly也允许部署grafana对peer节点的运行情况进行查看，来做到对私有网络运行情况的监控。最后我们也将在其上部署ipfs-cluster来对分布式存储系统来进行交互，以做到多媒体信息的存储并访问。

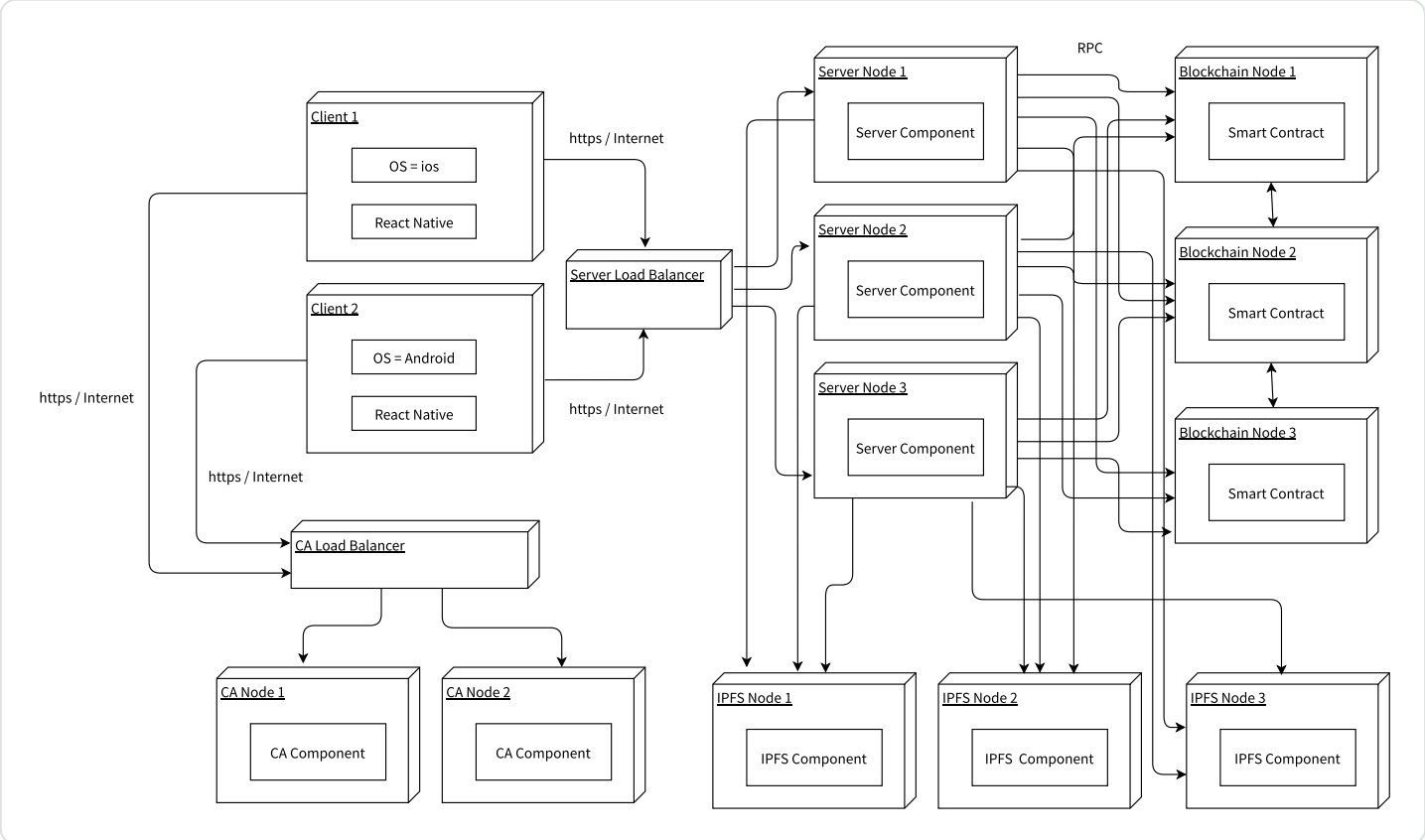
3.5 用户交互系统

用户交互系统通过使用RESTful API以及websocket同业务交互系统交互，来实现论坛帖子创建、消息收发，社交网络等一系列服务，同时也可通过firefly节点提供的公有链服务来提供面向实际应用背景的交易功能。

3.6 洋葱路由系统

所有的子系统之间都将用洋葱路由进行包裹，前端用户交互系统预计将使用libtor来对firefly的超级节点来进行访问，同时每个网络子系统的系统交互都将使用lokinet来进行路由。

4. 部署视图



5. 技术视图

前端App框架： React Native

后端框架： Hyperledger Fabric, IPFS cluster, libp2p

编程语言： Go / Typescript

数据库：IPFS + OrbitDB

容器编排： Docker / Docker Compose / Kubernetes

中间件：Hyperledger Firefly / Grafana

6. 数据视图（可选）

7. 核心算法设计（可选）

8. 质量属性的设计

1. 可扩展性：
    a. 使用容器以及智能合约的方式模块化的打包上线以做到业务逻辑的快速拓展
    b. 使用分布式的架构使得对服务感兴趣方来自主上线服务
2. 可靠性：
    a. Raft：全局都使用raft concensus来进行共识处理
    b. IPFS分布式存储CA数据以及用户发送信息以做到单一节点不可用时，用户仍然可以访问服务。
3. 可移植性：
    a. 使用容器编排帮助外部用户能够快速部署。
    b. 使用react native的前端以便于Android/IOS的跨平台开发。
4. 安全性：
    a. 使用洋葱路由对所有子系统之间的通信进行安全性增强
    b. 对于http使用tls来进行信道加密传输（待订）
    c. 对于特殊的私密信息将使用MTProto以及Chacha协议来做到端对端的对称加密