# 拉撒路 UTP

# V0.1

# 目录

# 1 历史

本节记录本文档的修改历史

| 版本 | 日期 | 作者 | 内容 |
|---|---|---|---|
| v0.1 | 2020 年 4 月 23 日 | 时空大神 | 初版本 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 2 系统结构

拉撒路系统的系统架构描述

| 编号 | 要件 | 内容 |
|---|---|---|
| 1 | 服务器 | 一个完整的系统需要（1+2n）+1 服务器：<br>一台作为 nexus-node 节点，1+2n 台作为 parallel-node 节点。 |
| 2 | 节点启动顺序 | 先启动 nexus-node 节点再启动 parallel-node 节点 |
| 3 | nexus-node 节点的确定 | 启动节点的时候根据配置文件确定。<br>Nexus_rotation_interval 时间周期后 Nexus 和 parallel-node 之间的切换， 从 parallel-node 候选池中中根据 reputation 值随机选出新的 nexus-node |
| 4 | 节点部署 | nexus-node 节点负责打包、签名、处理、广播交易。<br>parallel-node 节点负责对交易投票验证 |

## 3 系统状态

系统不同状态的描述

| | |
|---|---|
| 系统空闲 | 系统空闲时 nexus-node 节点和 parallel-node 节点处于监听 UDP 端口状态。nexus-node 节点监听是否有新的交易请求，parallel-node 监听 nexus-node 节点是否广播发送交易验证信息。<br>nexus-node 节点向 parallel-node 节点发送心跳包检测节点是否在线，将不在线的节点信息从 nexus-node 节点维护的在线节点表中移除。 |
| 系统运行 | 发送交易/验证交易 |

# 4 系统运行状态描述

详细描述系统运行状态的不同方式

## 4.1 交易处理

| 编号 | 用途 |
|---|---|
| 1 | nexus-node 创建交易信息，用交易发起方账户的私钥对交易进行签名，将交易信息序列化后通过 UDP socket 向 nexus-node 发送数据。 |
| 2 | nexus-node 节点通过 UDP 端口接收到交易信息 |
| 3 | 对接收到交易进行签名验证 |
| 4 | *对签名验证通过后的交易进行预处理。* |
| 5 | *交易预处理完毕后对交易结果验证，检测相关账户的余额是否正确。* |
| 6 | *对预处理成功后的交易的签名进行双重哈希运算。* |
| 7 | 将带有交易签名哈希的交易信息通过 UDP socket 广播到除了 nexus-node 节点之外的所有在线节点 |
| 8 | parallel-node 节点通过 UDP 端口接收从 nexus-node 节点发送过来的交易信息并进行处理 |
| 9 | 更新 parallel-node 账本的状态，使得和 nexus-node 账本状态一致 |
| 10 | parallel-node 节点对交易信息进行投票并将投票信息发送到 nexus-node 节点。 |
| 11 | nexus-node 节点通过 UDP socket 接收到 parallel-node 投票信息，检测账本状态和 nexus-nod 节点账本状态一致的 parallel-node 节点数量是否超过总的在线 parallel-node 节点数量的 2/3，如果超过 2/3 则 nexus-nod 节点将预处理交易的结果最终写入 nexus-node 节点上的永久性账本文件上。 |
| 12 | nexus-node 节点将最终确认的交易信息同步到 parallel-node 节点。 |

| | |
|---|---|
| 13 | parallel-node *节点从* nexus-node *节点拉取账本数据。* |

## 4.2 TPS 测速

| 编号 | 用途 |
|---|---|
| 1 | 生成相关配置文件 nexus-id.json、parallel-id.json 用于创建 Nexus/核心、Parallel/平行节点。 |
| 2 | 开启 TbkGiver 服务向所有测试节点发送 token |
| 3 | 在一台服务器上启动 Nexus/核心节点，Nexus/核心节点启动了 Fetch Stage、Sigverify Stage、Banking Stage、Write Stage 服务用于接收、发送、处理交易请求 |
| 4 | 两台服务器用作 Parallel/平行节点，Parallel/平行节点启动 Fetch Stage、Replicate Stage 服务对 nexus-nod 节点发送过来的 packet 进行处理 |
| 5 | 创建 50，000 个测试账户用于生成转账交易，给这些测试账户空投测试 token 用于发送交易。 |
| 6 | 向 Nexus/核心节点账户地址空投 50，000 个 token 用于向测试账户进行转账交易 |
| 7 | 检查各个账户余额是否充足 |
| 8 | 余额充足则在 Nexus/核心节点和测试账户间来回进行转账交易 |
| 9 | 启用多线程向 Nexus/核心节点发送交易 |
| 10 | 检测交易签名确认交易是否处理成功，计算交易发送到确认成功的时间（如果确认时间超过 3 mins 则丢弃交易） |
| 11 | 检测账户的余额是否正确 |
| 12 | 每间隔 1 秒钟计算一次 tps |

| | |
|---|---|
| 13 | 测试结束，统计计算转账交易测试期间内的最大 tps |

## 5 测试策略

| 编号 | 目标 | 过程 | 预期 |
|---|---|---|---|
| 1 | 生成 keypair 文件 | 创建以下文件：<br>nexus-id.json<br>parallel-id.json<br>client-id.json<br>用于存放节点 seed<br>并用于创建 seed-id.json | 文件生成 |
| 2 | 生成节点配置文件 | 创建 nexus.json 文件存放 nexus-id、address 信息 | 文件生成 |
| | | 创建 parrallet.json 文件存放 parallel-id、address 信息 | 文件生成 |
| 3 | 启动 TbkGiver 服务 | 运行 TbkGiver 服务用于向用户账户空投代币 | 启动成功 |
| 4 | **启动 Nexus/核心节点** | 根据 nexus.json 和 ledger 配置文件启动 CompleteNode 全节点，写日志到 log 文件<br>启动 CompleteNode 全节点<br>检测余额是否充足 余额不足则进行空投<br>检测 CompleteNode 全节点轮转状态 | 启动成功 |

|  |  |  |  |
|---|---|---|---|
| 5 | **启动 Proposer/提案节点** | 找到 Nexus/核心节点并从 Nexus/核心节点同步账本数据<br>启动 CompleteNode 全节点<br>检测余额是否充足 余额不足则进行空投<br>检测 CompleteNode 全节点轮转状态 | 启动成功 |
| 6 | **启动 BenchMarker/测速节点** | *进行 bench-tps 测试：*<br>1) 获取 client id , 网络地址，测试周期等信息<br>2) 选择 Nexus/核心节点<br>3) 找出网络上所有的节点<br>4) 创建随机数， 根据随机数生成 keypair<br>5) 空投 50_000 代币到 Nexus/核心节点， 空投 1 个代币到 barrier 地址<br>6) 多线程 transfer 交易<br>7) 查看节点余额是否不足<br>8) 余额充足则创建交易在节点和测试 carrier 地址之间进行 50_000 次转账<br>9) 多线程对交易进行发送、验证<br>10) 统计计算 tps | 启动成功 |