Radian 协议分析

注意: 以下内容来源于知乎, 我这里对照代码验证了一遍

1 Radiant的产品设计目标

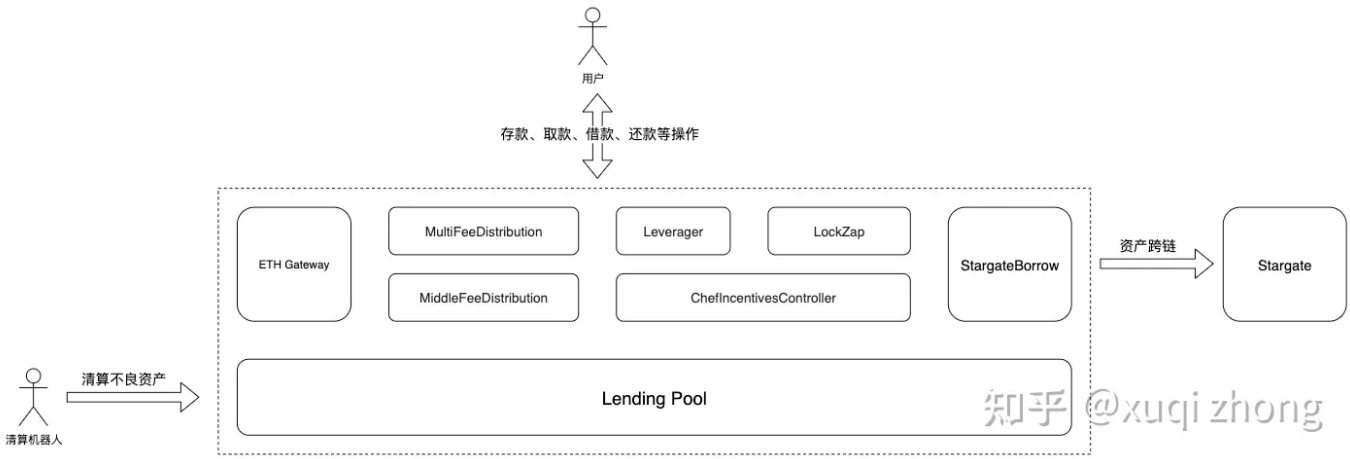
支持无缝跨链的去中心化借贷平台。

2 Radiant的核心竞争力

1. **去中心化**。Radiant的所有功能都是智能合约实现并且开源，任何人都可以无需KYC的情况下使用这款产品。
2. **数据公开透明**。Radiant的所有数据都是链上公开，任何人都可以方便地从链上获取radiant的营收数据。
3. **无缝跨链**。用户可以通过Radiant方便地进行资产跨链，并实现跨链抵押和借贷。
4. **高资金利用率**。通过Radiant的loop功能，可以实现杠杆（最高4倍），提高资金的年回报率。

3 Radiant的系统整体设计

系统架构图



系统各模块主要功能

- LendingPool: 平台核心逻辑，负责核心总账目的计算。
- ETHGateWay: 把ETH转为WETH，和把WETH转为ETH。
- MultiFeeDistribution: 分配平台奖励和借贷手续费。

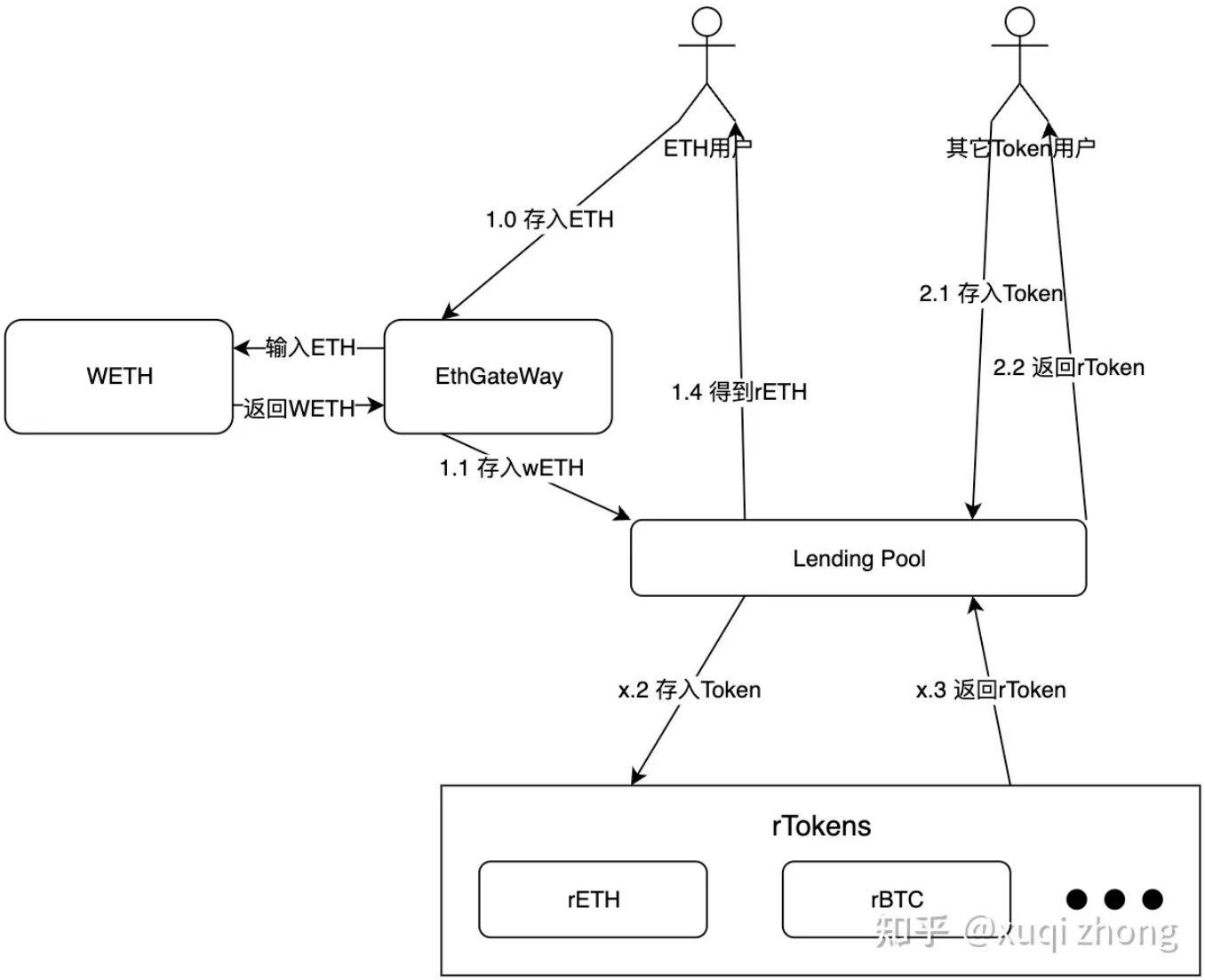
- MiddleFeeDistribution: 保存平台的借贷手续费，并把费用分配给用户。
- Leverager: 用户存款杠杆。
- LockZap: 锁定Zap资产的核心逻辑。
- ChefIncentivesController: RNDT奖励的分配计算逻辑。
- StargateBorrow: 跨链借贷逻辑。

4 Radiant核心业务逻辑解读

了解了Radiant系统设计框架，那么解析来就逐个解析它的核心业务流程。

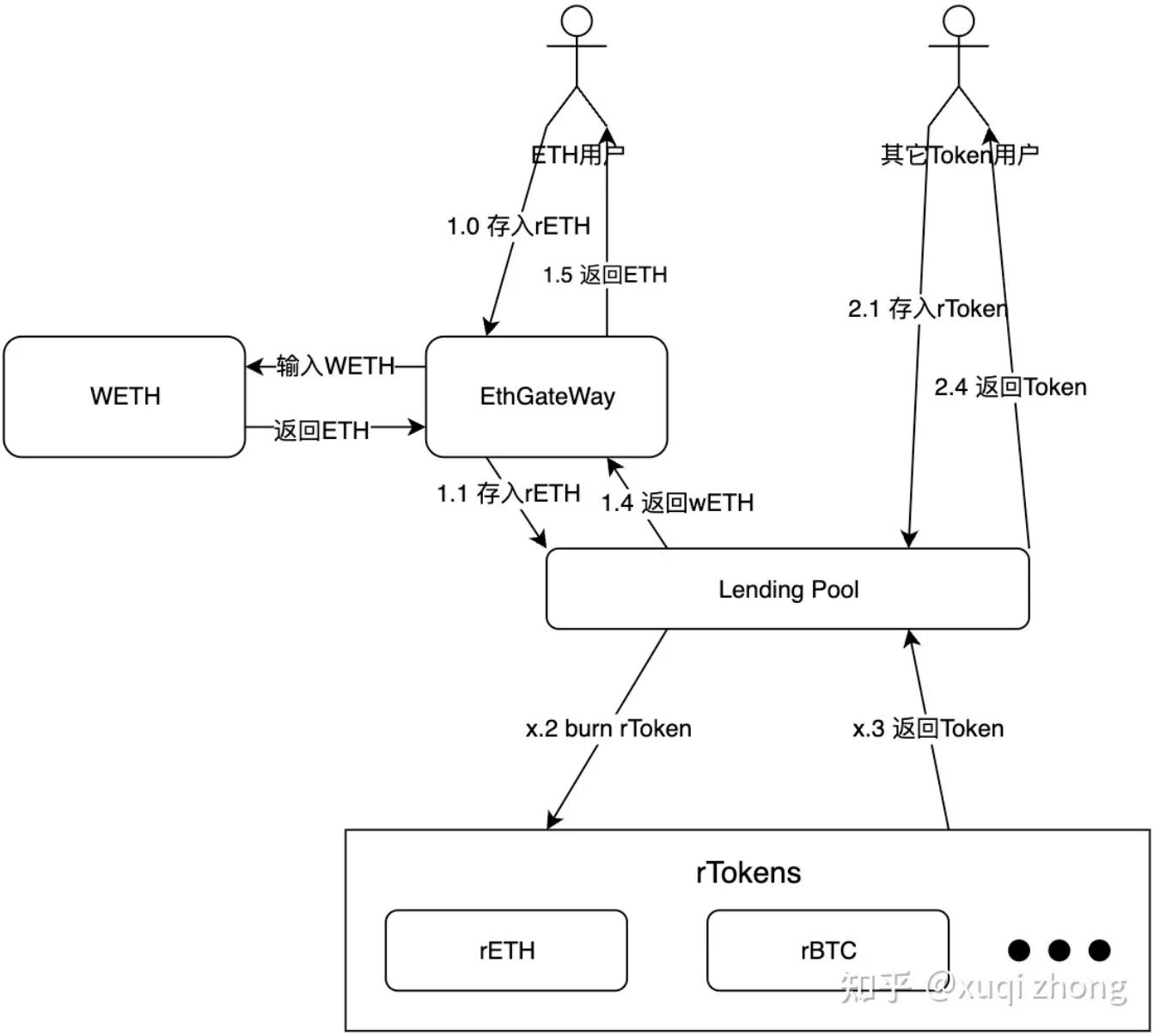
4.1 用户存取款业务逻辑

4.1.1 用户存取款业务流程



用户存款流程

1. 用户调用LendingPool合约，发起Token存入。
2. 如果是ETH，就先调用ETHGateway合约把ETH转成WETH。
3. ETHGateway合约再调用LendingPool合约执行后续流程。
4. LendingPool合约更新状态、存款利率和借款利率。
5. LendingPool合约把Token转入rToken合约。
6. rToken合约mint同等数量的rToken给到用户。
7. 用户按1:1得到rToken。



用户取款流程

1. 用户调用LendingPool合约，发起Token取款。
2. LendingPool合约更新状态、存款利率和借款利率。
3. LendingPool合约调用rToken合约burn rToken。
4. rToken合约按照1:1把同等数量Token给到用户。
5. 用户得到Token。
6. 如果是ETH，就通过ETHGateway合约把ETH转成WETH。
7. ETHGateway合约再把WETH转给用户。

4.1.2 用户存取款业务解读

从业务流程中可以看出，LendingPool合约负责所有的存取款业务。 用户存入Token后，会得到rToken；燃烧rToken后会得到Token。 Token和rToken是1:1兑换的，用户的rToken会随着时间增长。

rToken增长计算公式

```
rToken数量 = 当前rToken数量 * 时间间隔 * 存款利率
```

存款利率计算计算公式

```
存款利率 =（变化利率贷款总额 * 变化利率 + 固定利率贷款总额 * 平均固定利率）* 资金利用率 /
（变化利率贷款总额 + 固定利率贷款总额）
```

资金利用率

```
资金利用率 =（变化利率贷款总额 + 固定利率贷款总额）/ 存款总额
```

相关合约地址

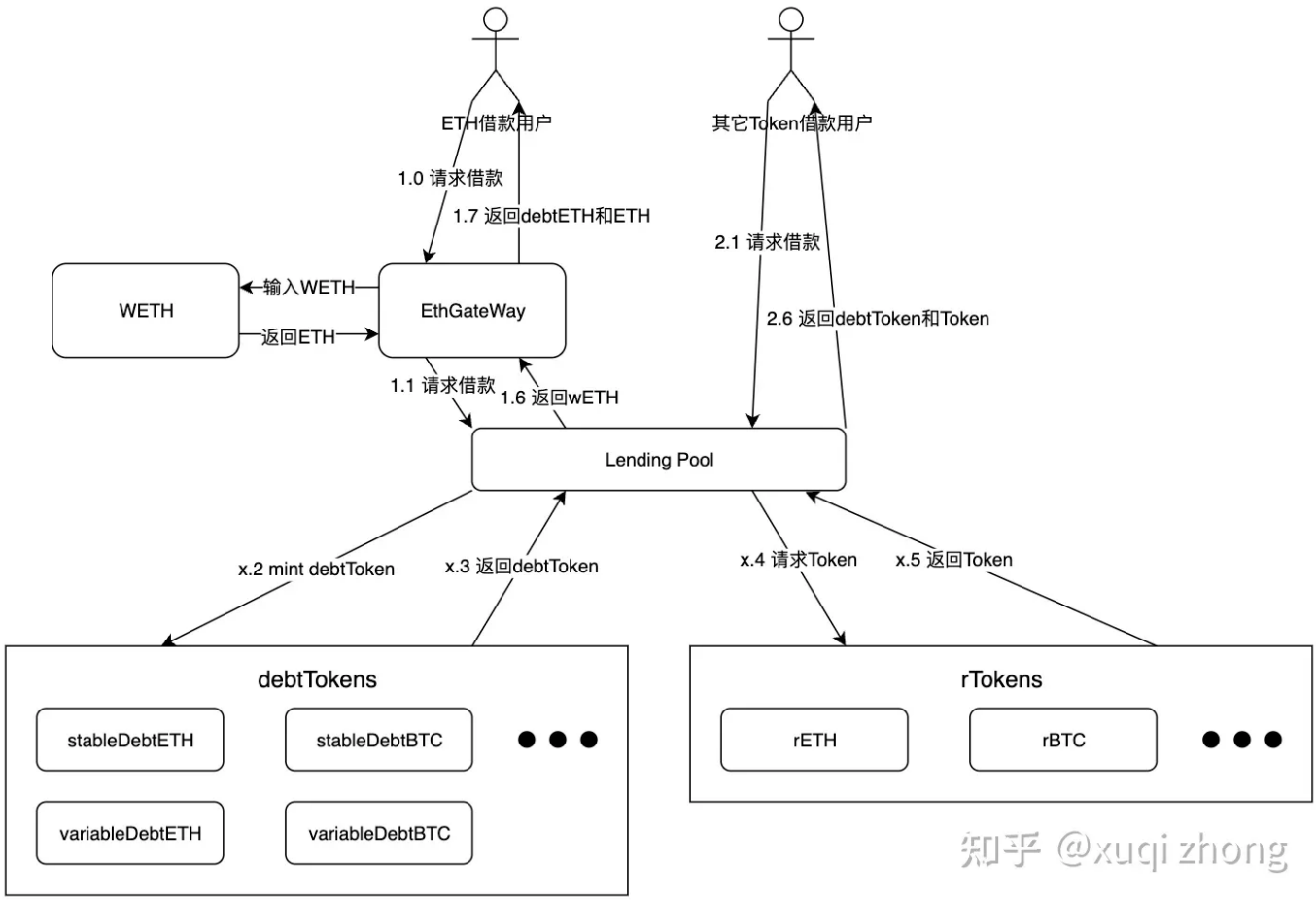
合约名称	合约地址
LendingPool	0xf4b1486dd74d07706052a33d31d7c0aafd0659e1
ETHGateway	0xbb5ca40b2f7af3b1ff5dbce0e9cc78f8bfa817ce

相关交易操作交易Hash

操作名称	交易Hash
ETH存款	0x157f355b2163e149d6c051486f3f7d7c44443dcb8446521f097585603712f382
Token存款	0x710cffe98f9af5e7b0adc931a14d0af2fefc95917d00e33e41f0300d823be918
ETH取款	0xdfadf35940600854314318e0d8269ec720ae1026119ff0997c3fa53b0bf9130c
Token取款	0xe3dd4244e62c37a2e7f09db438eb747c9ce29008e7f0fd7cce4eae8bc7966a20

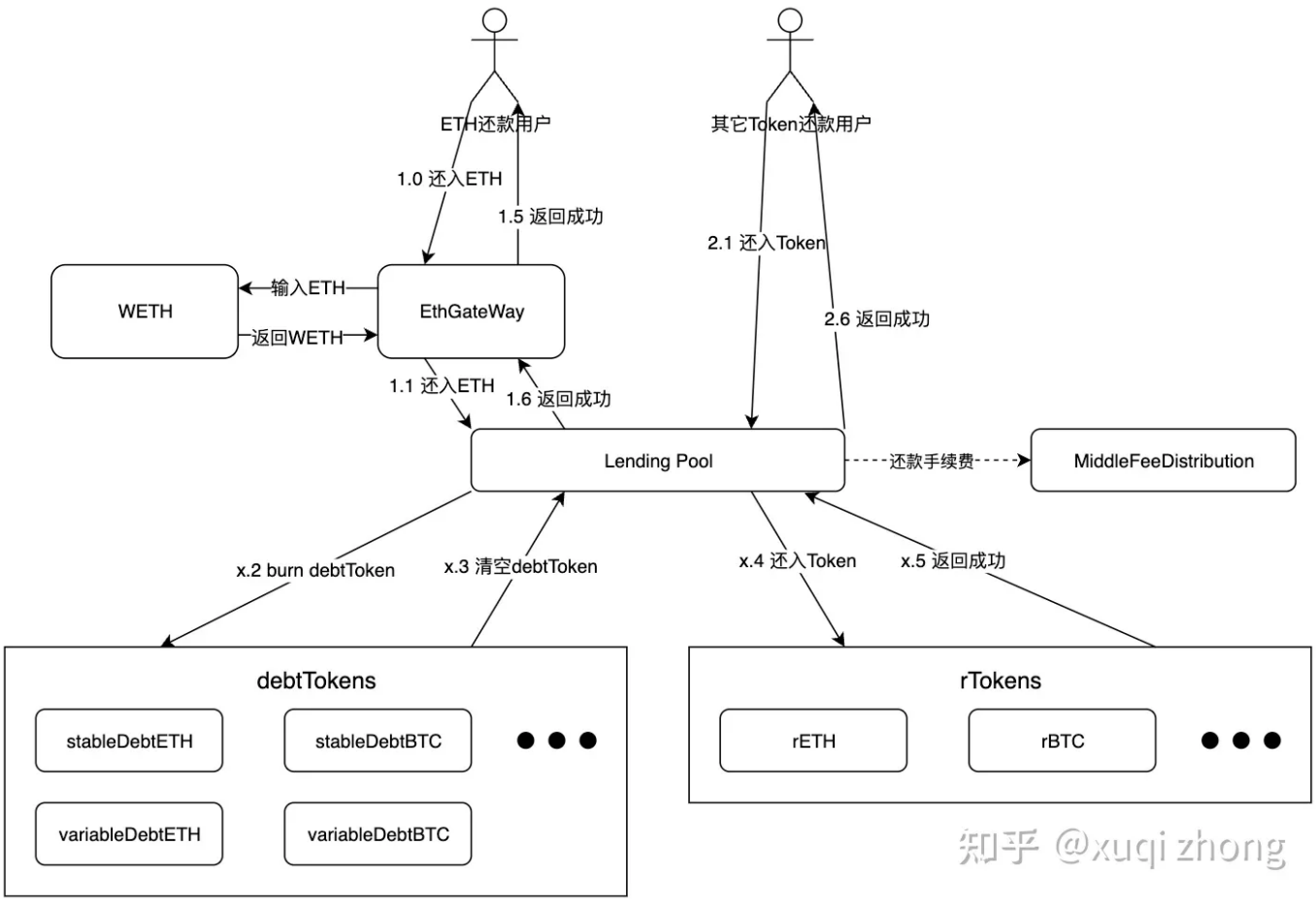
4.2 用户借还款业务逻辑

4.2.1 用户借还款业务流程



用户借款流程

1. 用户调用LendingPool合约，发起Token借款。
2. 如果是ETH，就先ETHGateway合约再调用LendingPool合约执行后续流程。
3. LendingPool合约检查用户抵押品是否足够满足借出Token的数量。
4. LendingPool合约更新状态。
5. LendingPool合约调用xxDebtToken合约，mint出xxDebtToken到用户地址下。
6. 此处需要根据用户借贷的参数选择stableDebtToken（稳定利率借贷）或者variableDebtToken（变化利率借贷）。
7. LendingPool合约更新存款利率和借款利率。
8. LendingPool合约调用rToken合约，借出Token到用户地址到。
9. 用户得到Token。
10. 如果是ETH，ETHGateway合约会把得到的wETH换成ETH并转给用户地址。



用户还款流程

1. 用户调用LendingPool合约，发起Token还款。
2. 如果是ETH，就先ETHGateway合约再调用LendingPool合约执行后续流程。
3. LendingPool合约检查用户的请求是否正确。
4. LendingPool合约更新状态。
5. LendingPool合约调用xxDebtToken合约，burn掉用户地址下的xxDebtToken。
6. 此处需要根据用户借贷的参数选择stableDebtToken（稳定利率借贷）或者variableDebtToken（变化利率借贷）。
7. LendingPool合约更新存款利率和借款利率。
8. LendingPool合约把用户地址下的Token转移到rToken合约。

4.2.2 用户借还款业务解读

从业务流程中可以看出，LendingPool合约负责所有的借还款业务。 在借款业务中，用户需要mint xxDebtToken，代表用户的借贷金额；相反，在还款业务中，用户需要burn xxDebtToken，代表解除自身的债务。 xxDebtToken是一类特殊的ERC20 token，它只能被mint和burn，无法被transfer，同时也会随时时间增长。

stableDebtToken和variableDebtToken的增长计算公式

```
stableDebtToken数量 = 当前stableDebtToken数量 * 时间间隔 * 稳定贷款利率

variableDebtToken数量 = 当前variableDebtToken数量 * 时间间隔 * 变化贷款利率
```

稳定贷款利率和变化贷款利率的增长计算公式

$$\text{稳定贷款利率} = \begin{cases} 10\% \times \dfrac{\text{资金利用率}}{\text{最佳利用率}} & \text{if 资金利用率} <= \text{最佳利用率} \\ 10\% + 95\% \times \dfrac{\text{资金利用率} - \text{最佳利用率}}{\text{超额利用率}} & \text{if 资金利用率} > \text{最佳利用率} \end{cases}$$

$$\text{变化贷款利率} = \begin{cases} 13\% \times \dfrac{\text{资金利用率}}{\text{最佳利用率}} & \text{if 资金利用率} <= \text{最佳利用率} \\ 13\% + 95\% \times \dfrac{\text{资金利用率} - \text{最佳利用率}}{\text{超额利用率}} & \text{if 资金利用率} > \text{最佳利用率} \end{cases}$$

不同Token的最佳利用率和超额利用率

Token名称	最佳利用率	超额利用率
wBTC	70%	30%
USDT	62%	38%
USDC	61.5%	38.5%
DAI	61.5%	38.5%
wETH	70%	30%

相关合约地址

合约名称	合约地址
LendingPool	0xf4b1486dd74d07706052a33d31d7c0aafd0659e1
ETHGateway	0xbb5ca40b2f7af3b1ff5dbce0e9cc78f8bfa817ce
MiddleFeeDistribution	0xE10997B8d5C6e8b660451f61accF4BBA00bc901f

相关交易操作交易Hash

操作名称	交易Hash
ETH借款	0xd1e69e7949ad40b17996297124d19705acc74694a95c9285a3b61d4b07c5c20c
Token借款	0x108c3a9b16e56cb505a4d67f5acf70203cb3e4fff682b6492b7df71eb051e481
ETH还款	0xf171302948269c06b796f98b17c10a1496a8c71ed42b1cccaf61d696af1df12e
Token还款	0xfc8658a7530102ff5ac9731c9e2856f228f15a04585f659bc3290554782c2fa8

4.3 用户loop业务逻辑

4.3.1 用户loop业务流程

1. 用户调用Leverager合约发起loop操作。
2. Leverager合约接收用户Token，并进行存款操作。
3. 循环如下操作n次。
4. Leverager合约把用户存款作为抵押，从LendingPool合约借出一定数量的Token。
5. Leverager合约把借出的Token再次存入LendingPool合约。
6. 以用户存款总额的5%，借出ETH并进行Zap。

4.3.2 用户Zap资产业务解读

用户loop业务其实就是循环存款和贷款的过程，即以存款为抵押贷款，并将贷款再次存款。 因为存款的抵押的借贷率为75%，因此最多循环4次就会达到极限。 循环借贷会降低用户资产的健康度，在市场不稳定的情况可能会导致爆仓被清算。

相关合约地址

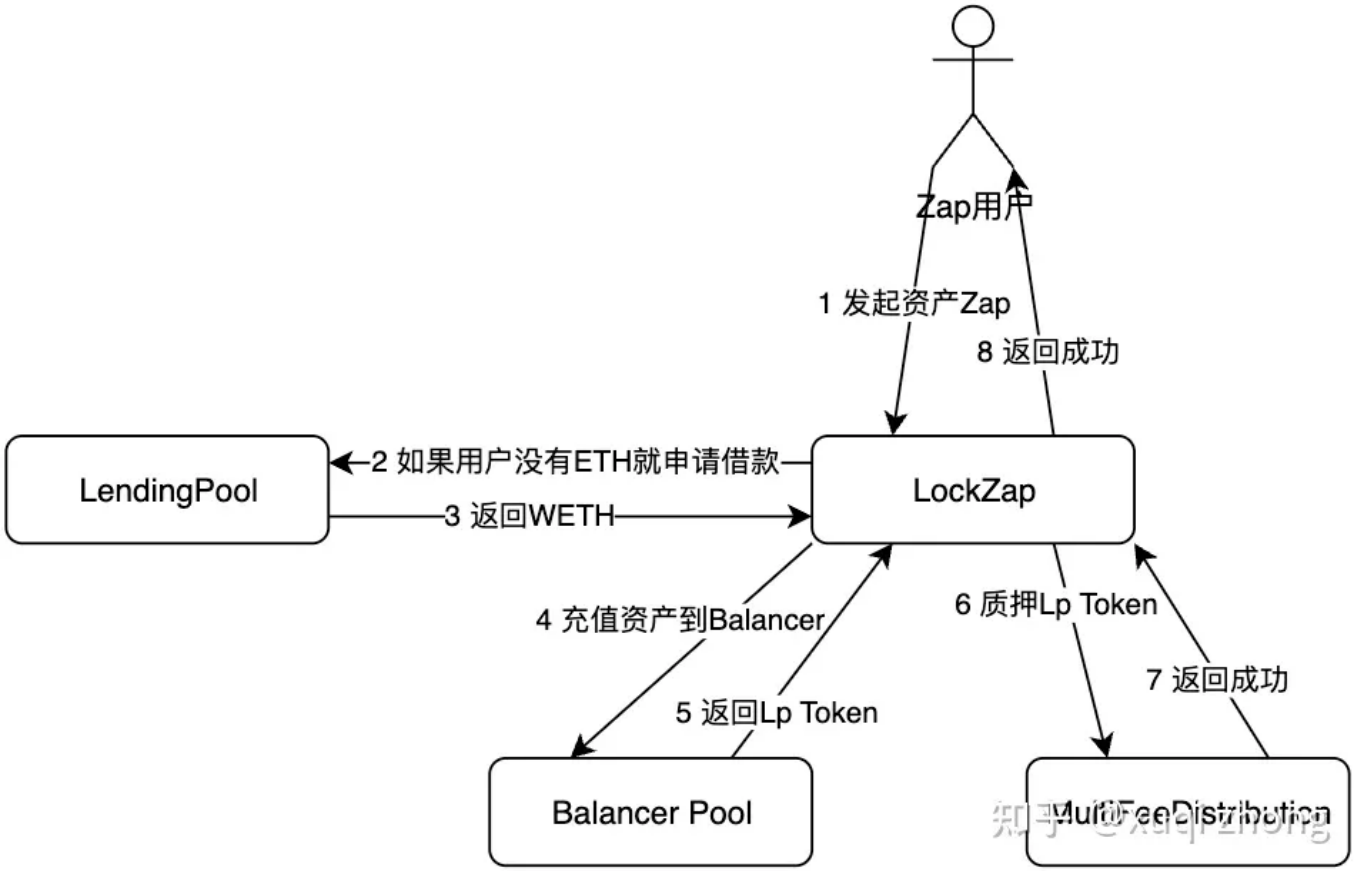
合约名称	合约地址
Leverager	0x196bF3A63c50BcA1Eff5A5809B72DFc58F0C2c1a
LendingPool	0xf4b1486dd74d07706052a33d31d7c0aafd0659e1

相关交易操作交易Hash

操作名称	交易Hash
用户Loop Token	0x350331fa82279cd44ba605540ae64a31ae76626487ee1e835be906797454e094
用户Loop ETH	0x8f86e873e9a0bc905c46f6b911462f58b21adc94ada29f5d683e1a8be2c1d6cc

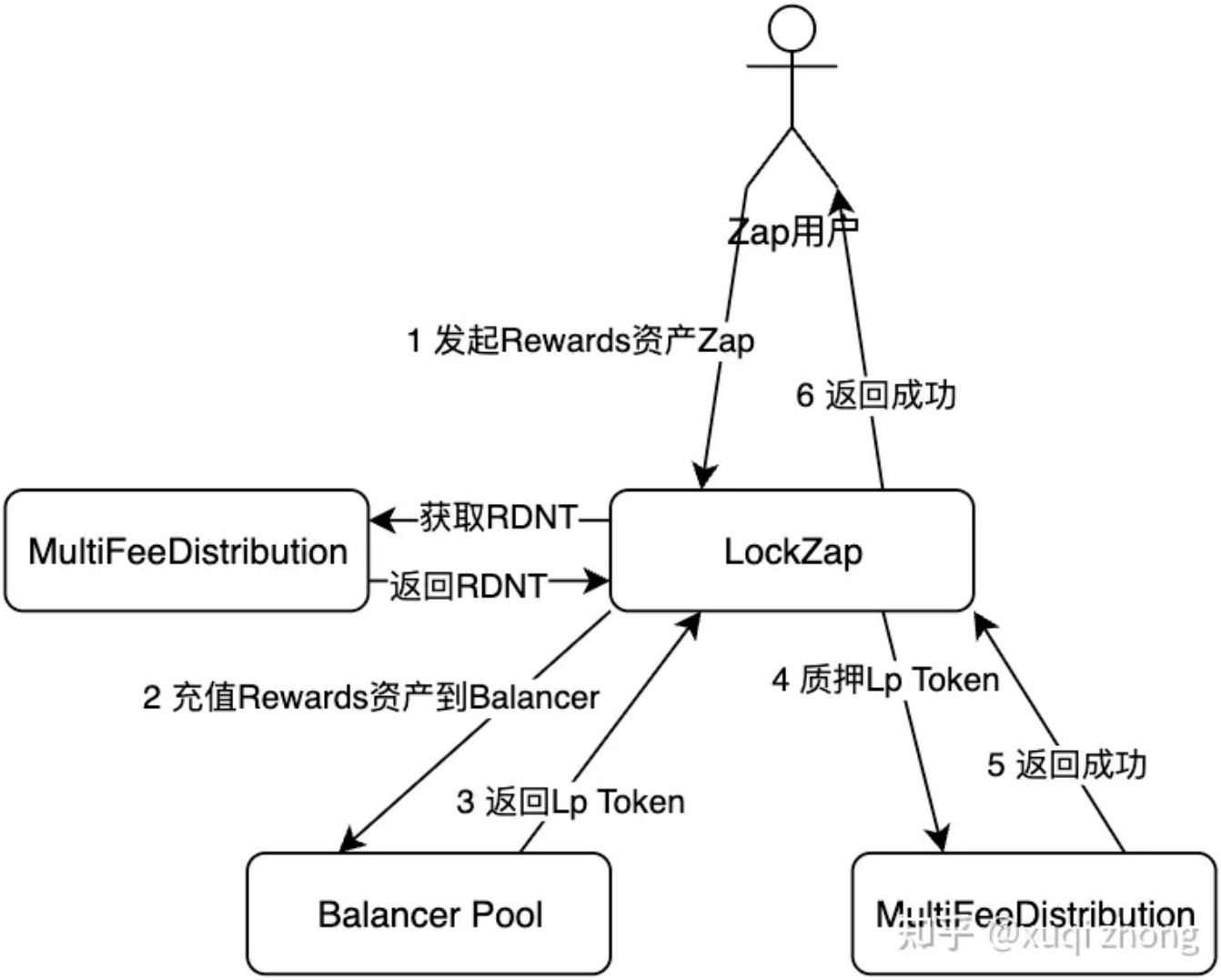
4.4 用户Zap资产业务逻辑

4.4.1 用户Zap资产业务流程



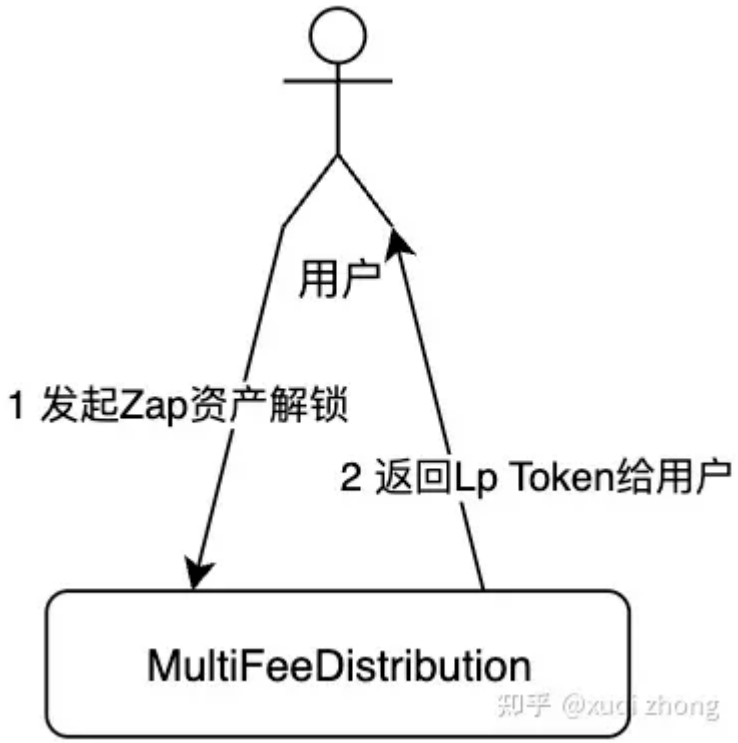
用户Zap资产流程

1. 用户调用LockZap合约发起资产Zap操作。
2. 如果用户没有传入ETH，就调用LendingPool合约借出ETH。
3. LockZap合约把用户资产充值到Balancer Pool，得到LP Token。
4. LockZap合约把LP Token质押到MultiFeeDistribution合约中。



用户Zap Reward资产流程

1. 用户调用LockZap合约发起资产Zap操作。
2. LockZap合约调用MultiFeeDistribution合约获取用户得到的RDNT代币资产
3. LockZap合约把用户RDNT代币资产充值到Balancer Pool，得到LP Token。
4. LockZap合约把LP Token质押到MultiFeeDistribution合约中。



用户UnZap资产流程

1. 用户调用MultiFeeDistribution合约发起资产UnZap操作。
2. MultiFeeDistribution合约计算用户已经到期的Zap资产数量，即LP Token数量
3. MultiFeeDistribution合约把LP Token返还给用户。

4.4.2 用户Zap资产业务解读

从业务流程可以看出，Zap就是把用户的资产放到Balancer的ETH-RDNT Pool中，给ETH与RDNT之间的交易提供流动性。 同时，LockZap合约会把LP Token质押到MultiFeeDistribution合约中，根据用户的质押时长不同可以享受到不同的手续费收益。 另外，如果Zap资产的总价值超过用户存款总价值的5%，用户还可以获得RDNT的增发奖励。 锁仓有利于稳定RDNT的币价，减少RDNT的抛压。

相关合约地址

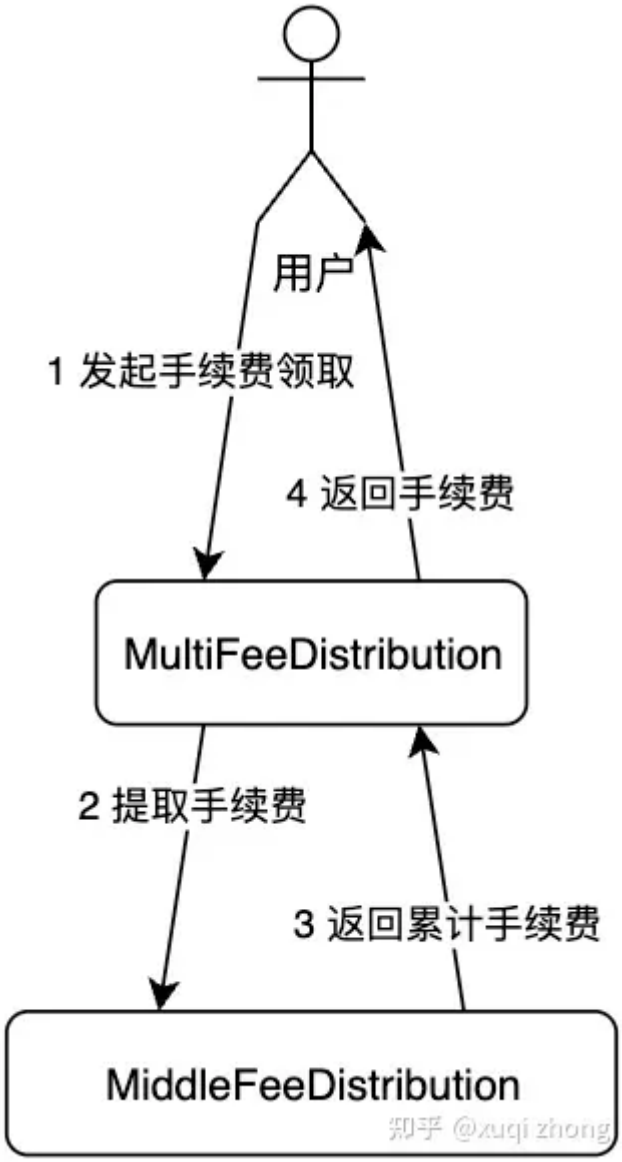
合约名称	合约地址
LockZap	0x8991c4c347420e476f1cf09c03aba224a76e2997
LendingPool	0xf4b1486dd74d07706052a33d31d7c0aafd0659e1
MultiFeeDistribution	0x76ba3ec5f5adbf1c58c91e86502232317eea72de

相关交易操作交易Hash

操作名称	交易Hash
用户Zap资产	0xed757145e3fba02a1350b88c54dc14e80a8d33133673e20d790a4cf8d7bc624e
用户Zap Rewards资产	0x683127f8cb1ba9b2c47b6b288f71f762804eb654f47a7d1c4824fde2cf8bc540
用户UnZap资产	0x6d1bc88d6bd3317d2ad75fba0bbbe3271f1e3118198054e5bdcdbbe9f34e1c5a

4.5 用户领取手续费业务逻辑

4.5.1 用户领取手续费业务流程



领取手续费流程

1. 用户调用MultiFeeDistribution合约发起手续费领取。
2. MultiFeeDistribution合约根据用户锁定的Zap资产数量，计算用户的手续费所得。
3. MultiFeeDistribution合约调用MiddleFeeDistribution合约获取累计手续费。
4. MultiFeeDistribution合约把手续费转给用户所在地址。

4.5.2 用户领取手续费业务解读

从流程中可以看出手续费主要保存在MiddleFeeDistribution合约中，并由MultiFeeDistribution合约分配给用户。

MiddleFeeDistribution合约抽取手续费计算公式

```
总手续费 =（变化贷款总额 x 变化贷款利率+固定贷款总额 x 平均固定贷款利率）x 时长 x 手续费费率
```

其中，手续费费率由管理员设定。

用户手续费权重计算公式

```
手续费 = 总手续费 x 质押时长系数 x 用户资产总额平均质押时长系数 x 资产总额
手续费 = 总手续费 x 平均质押时长系数x 总Zap资产总额质押时长系数 x 用户Zap资产总额
```

其中，质押时长系数根据用户的质押时长不同而不同，如下表所示

质押时长	1个月	3个月	6个月	12个月
质押时长系数	1	4	10	25

相关合约地址

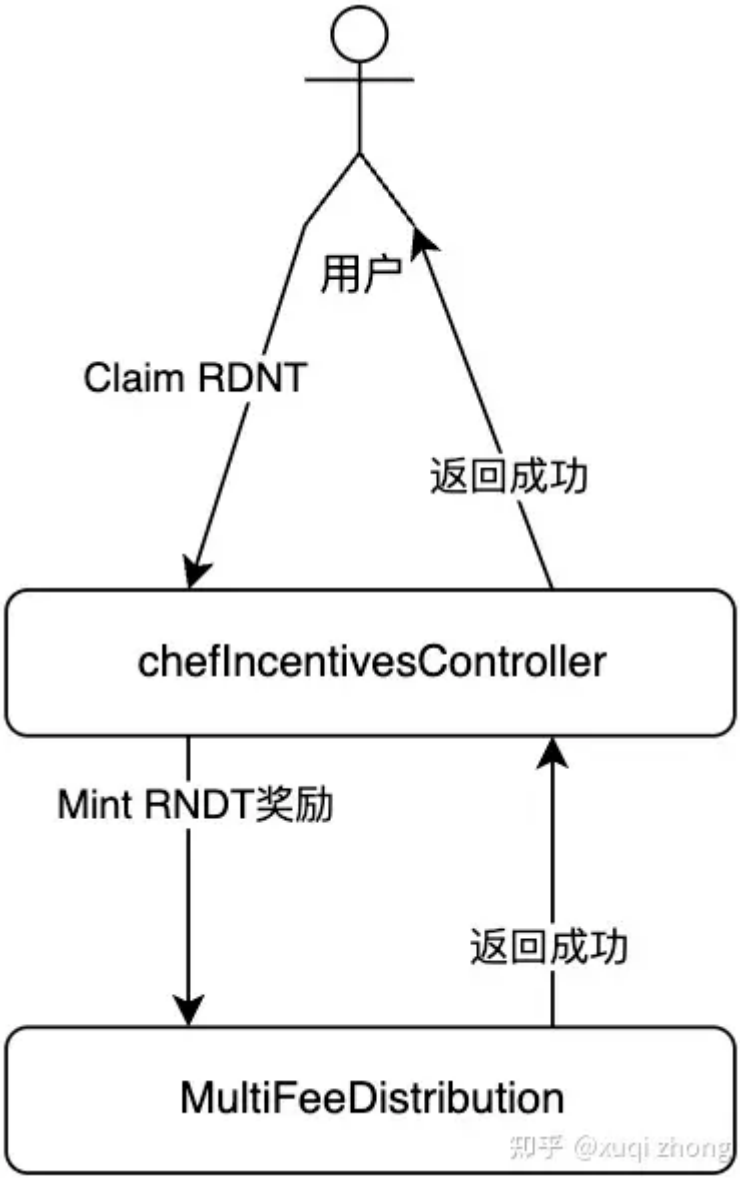
合约名称	合约地址
MultiFeeDistribution	0x76ba3ec5f5adbf1c58c91e86502232317eea72de
MiddleFeeDistribution	0xE10997B8d5C6e8b660451f61accF4BBA00bc901f

相关交易操作交易Hash

操作名称	交易Hash
领取交易手续费	0x069725532db0d6ec70d62cfd602db98a48efc5be38bc7f757fe055efdd187633

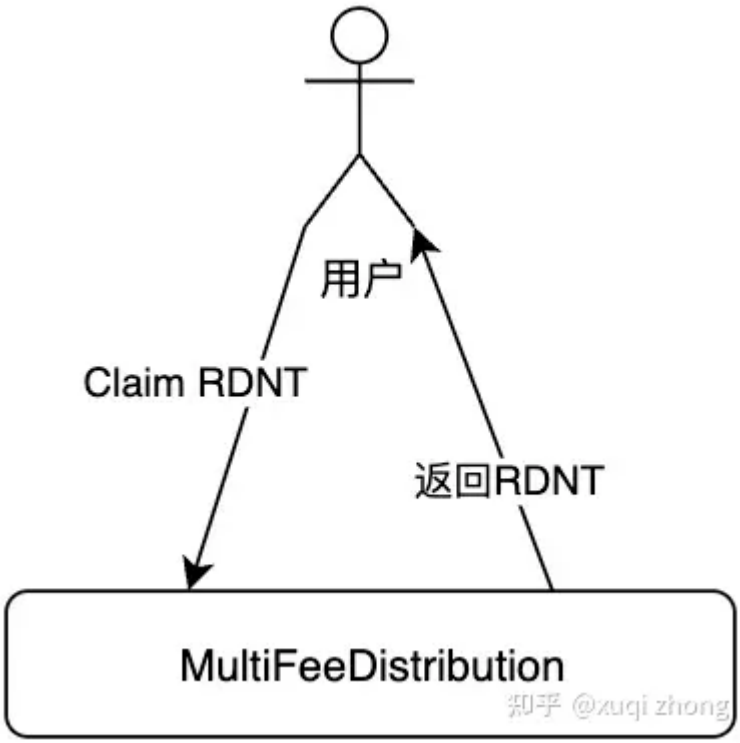
4.6 用户领取RDNT奖励业务逻辑

4.6.1 用户领取RDNT奖励业务流程



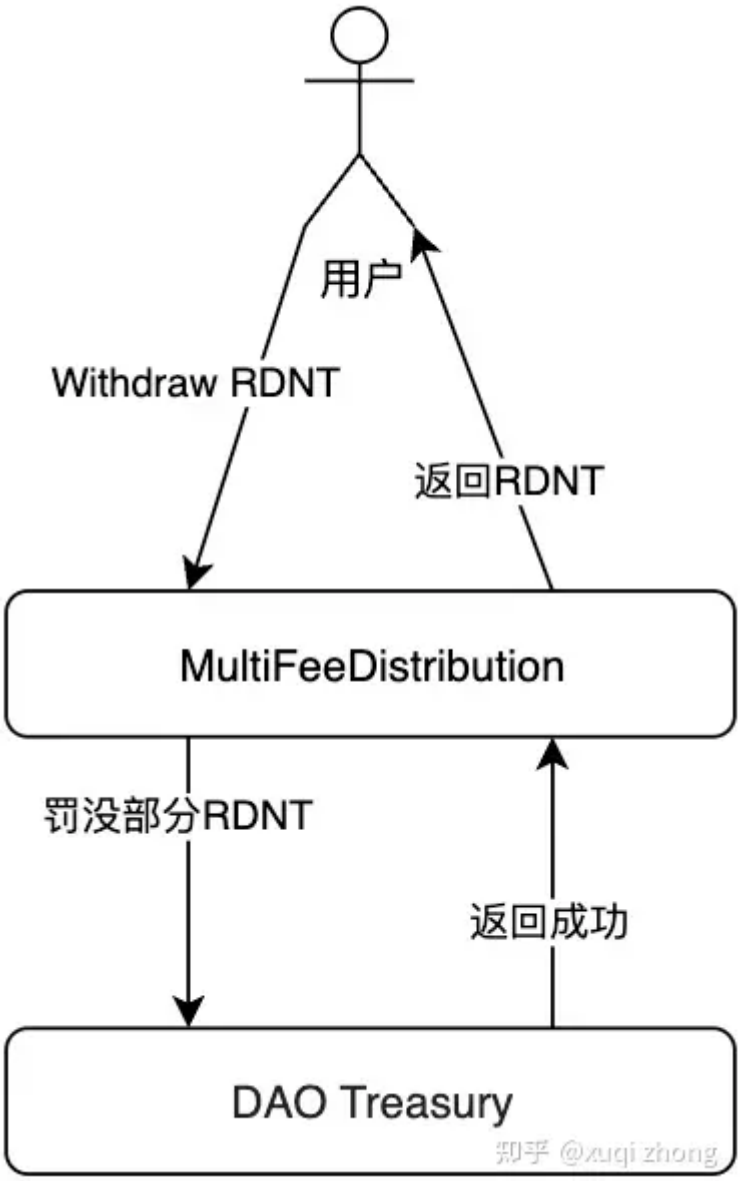
用户Claim RDNT流程

1. 用户调用ChefIncentivesController合约发起RDNT Claim。
2. ChefIncentivesController合约判断用户是否具有领取资格，即Zap资产总额大于等于存款总额的5%。
3. ChefIncentivesController合约计算用户的RDNT数量，并发送到MultiFeeDistribution合约。
4. MultiFeeDistribution合约锁定用户的RDNT，时长为90天，即90天后可以可以领取RDNT。



用户Withdraw RDNT流程

1. 用户计算到期的RDNT奖励数量（90天到期）。
2. 用户调用MultiFeeDistribution合约发起RDNT Withdraw，领取数量小于等于到期的RDNT奖励数量。
3. MultiFeeDistribution合约根据用户的领取数量，从早到晚依次领取RDNT。



用户提前Withdraw RDNT流程

1. 用户调用MultiFeeDistribution合约发起RDNT提前Withdraw。
2. MultiFeeDistribution合约计算用户的RDNT Token奖励数量和罚没RDNT Token奖励数量。
3. MultiFeeDistribution合约把用户的RDNT Token返还给用户。
4. MultiFeeDistribution合约把RDNT Token罚没数量传到DaoTreasury合约。

4.6.2 用户领取RDNT奖励业务解读

从流程中可以看出，ChefIncentivesController负责RDNT的分配资质检验，并把RDNT发送到MultiFeeDistribution合约。 MultiFeeDistribution合约负责RDNT的锁定和领取逻辑。

用户RDNT奖励计算公式

```
RDNT奖励数量 = 用户存款总额 x RDNT奖励利率 x 时长
```

其中，RDNT奖励利率由管理员来设定。

用户提前领取RDNT罚没百分比计算公式

```
罚没RNDT百分比 = 25% + 65% x (1 – 质押时长 / 90天)
```

罚没百分比质押时长天时长

相关合约地址

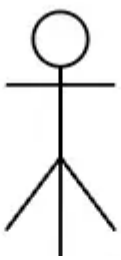
合约名称	合约地址
ChefIncentivesController	0xebc85d44cefb1293707b11f707bd3cec34b4d5fa
MultiFeeDistribution	0x76ba3ec5f5adbf1c58c91e865022323 17eea72de
DaoTreasury	0x76ba3ec5f5adbf1c58c91e865022323 17eea72de

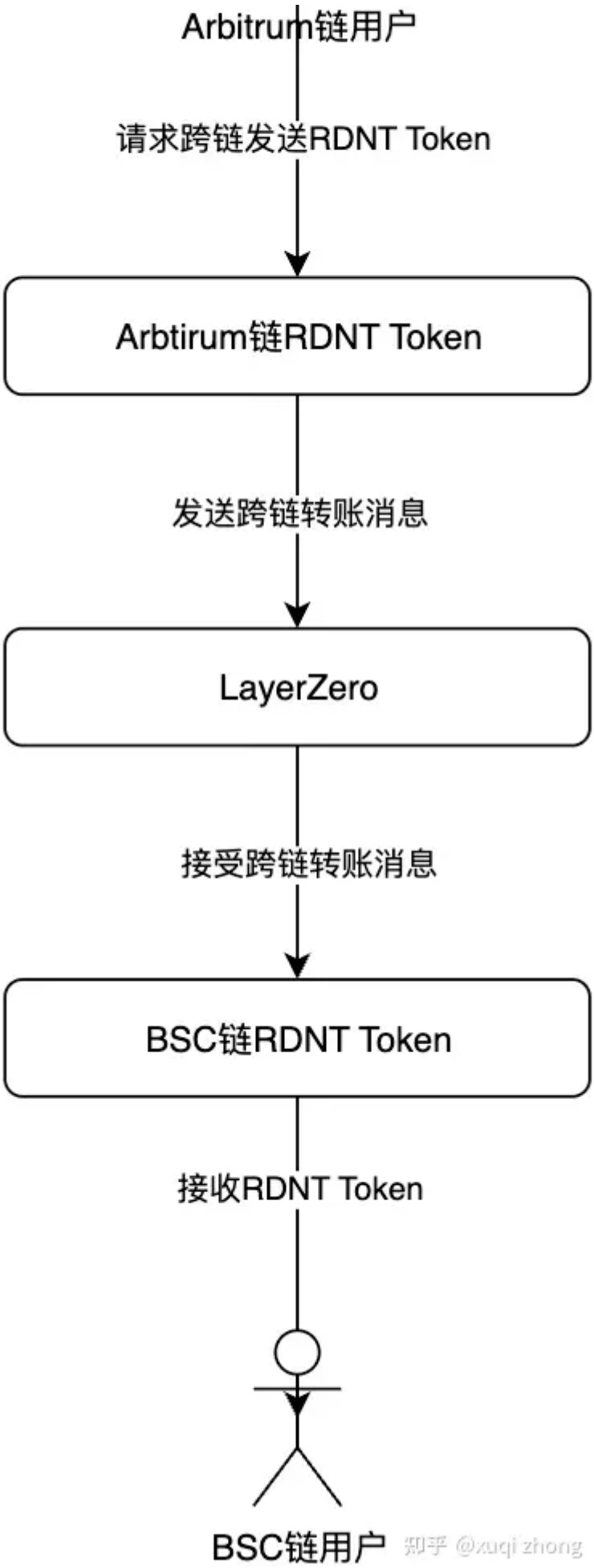
相关交易操作交易Hash

操作名称	交易Hash
用户Claim RDNT	0xf78f720536d1fd92b0f281332c371dd292dd267660a0febaf7a0438b33d7bce2
用户提前Withdraw RDNT	0x84b69c45e7f21fa2c5ffd78a3c8dd7e038a05ec3d9f6aee3b785d21517c576a9

4.7 用户跨链转账RDNT业务逻辑

4.7.1 用户跨链转账RDNT业务流程





RDNT跨链转账流程

1. 用户调用RDNT Token合约发起RDNT Token跨链转账。
2. RDNT Token合约burn掉RDNT Token，并发送跨链转账消息到LayerZero。
3. LayerZero把跨链转账消息发送的目标链合约。
4. LayerZero目标链合约mint出RDNT Token，mint RDNT Token并转给用户在目标链的地址下。

4.7.2 用户跨链转账RDNT业务解读

RDNT的主要跨链逻辑是基于LayerZero来实现跨链功能，RDNT继承了LayerZero的OFT技术实现。 这块技术细节后续会专门出一期来解读，欢迎大家持续关注。

相关合约地址

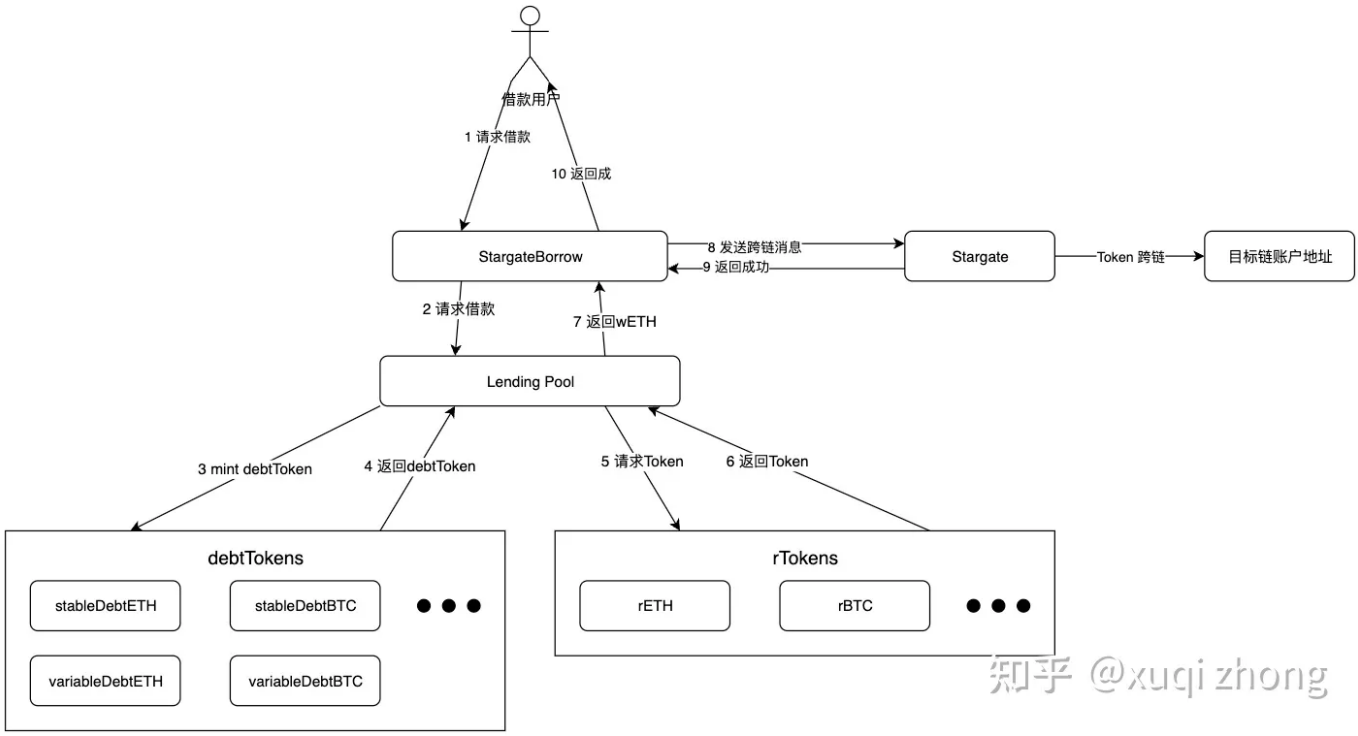
合约名称	合约地址
RDNT Token	0x3082CC23568eA640225c2467653dB90e9250AaA0

相关交易操作交易Hash

操作名称	交易Hash
用户跨链转账RDNT	0x2383e3bd250d5836f1a703d38d6e0fe05913c7fddb050aab7a8987144d23da74
目标链接收RDNT	0x777045e650fef686803f8adcb4df79671f31db946a987287a733085be97d54bf

4.8 用户跨链借贷业务逻辑

4.8.1 用户跨链借贷业务流程



跨链借款流程

1. 用户调用StargateBorrow合约发起跨链借贷请求。
2. StargateBorrow合约调用LendingPool合约，发起Token借款。
3. LendingPool合约检查用户抵押品是否足够满足借出Token的数量。
4. LendingPool合约更新状态。
5. LendingPool合约调用xxDebtToken合约，mint出xxDebtToken到用户地址下。
6. 此处需要根据用户借贷的参数选择stableDebtToken（稳定利率借贷）或者variableDebtToken（变化利率借贷）。
7. LendingPool合约更新存款利率和借款利率。
8. LendingPool合约调用rToken合约，借出Token到用户地址到。
9. StargateBorrow通过Stargate跨链桥进行跨链转账。
10. 目标链用户得到Token。

4.8.2 用户跨链借贷业务解读

跨链借贷和正常借贷的过程一样，区别就是就是最后会把借出的Token通过跨链桥转到目标链的用户地址下。

相关合约地址

合约名称	合约地址
StargateBorrow	0x9441fcd3e538a84e122ac6ffe3c07417cbe51dc9
LendingPool	0xf4b1486dd74d07706052a33d31d7c0aafd0659e1
MiddleFeeDistribution	0xE10997B8d5C6e8b660451f61accF4BBA00bc901f

相关交易操作交易Hash

操作名称	交易Hash
用户跨链借贷	0xbd1c371da45bcabb556b8f1018187c5e0882dd8d044a909b49e8f383c5cf7b85
目标链接收Token	0x10212177f45b96a35517b3196a017c23ca8b9675d422dc3931356db9ea9b4546

5 总结

作为用户来说，Radiant项目如下几个方面优势，可能是它的核心竞争力。

1. 项目核心代码几乎完全复制了Aave，因此安全性上面是可以保证的。
2. Radiant设计了一套代币经济模型，激励用户持续给平台提供流动性。
3. 基于LayerZero的技术，实现了资产跨链借贷的功能。

但是它的劣势，也非常明显。

1. 目前平台激励完成基于Radiant的增发奖励，无法长期维持。
2. 跨链技术基于LayerZero，自身并没有太多的技术创新。

但是抛开技术面分析，从当前来看Radiant在Arbitrum上面的TVL已经超过了Aave，可见市场对Radiant的认可度。