Aurora

去中心化的以太坊 Layer 2

NEAR

连接区块链与互联网世界

near.org

简介

- PoS共识机制，Layer 1区块链平台
- 追求极致的用户体验、开发体验
- 分片扩容
- 跨链互操作

$2B
市值

100k
链上日交易量

8k+
链上日活跃用户量

180k
链上账户数量

（自从彩虹桥上线已提升200%）

NEAR

NEAR Collective 团队背景

分布式：超过100人的全职贡献者，分布在10个国家

世界一流：4 次ACM ICPC编程大赛全球冠军、多枚奖牌，12次决赛经历; Google Code Jam冠亚军；TopCoder冠军

行业经验：谷歌、脸书、微软、MemSQL、Consensys

创业经验：11位连续创业者


MetaStable


ACCOMPLICE


a16z


ELECTRIC CAPITAL


PANTERA


Ampilfy PARTNERS


MULTI
COIN
CAPITAL


a_capital


coinbase | Ventures


IDEO
CO
LAB


SCALAR CAPITAL


XPRING


homebrew


Baidu.ventures
百度风投


SVAngel


IOSG
Venture


GREEN
FIELD ONE


arrington
XRP CAPITAL


FABRIC
VENTURES


Libertus Capital


BLOCKCHANGE


DISTRIBUTED GLOBAL

多链互操作的新时代

Aurora (EVM runtime)

以太坊L2
面向以太坊现有的开发者、用户，DeFi玩家

NEAR (Native Runtime)

开放网络平台
连接区块链和互联网世界的桥梁

Octopus Network (Substrate)

应用链网络
针对Substrate开发者

NEAR Native 原生生态

- 夜影动态分片，横向扩容
- 跨分片交易仅需一个区块
- 隐藏区块链，为用户提供熟悉的体验
  - 用户无需考虑交易（支付gas费）
  - 无需了解公钥/私钥或备注词
  - 定制化的使用权限
- 生态：开放金融，NFT，社交网络等等

提供不逊于当今互联网的性能和使用体验
成为人人可用的开放网络平台

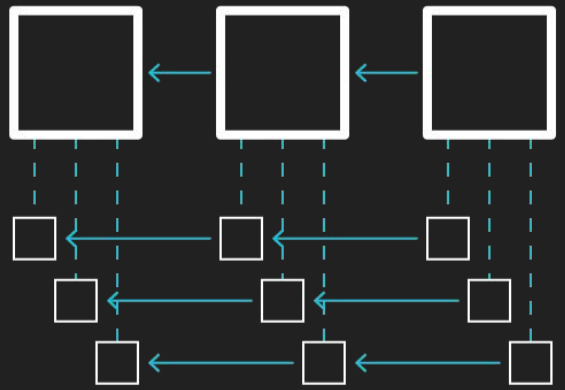
扩容 - 夜影分片协议

- 扩容：50x ETH 1.0 / 分片
  - Gas比ETH便宜1000-10000倍
- 动态再分片
  - 随网络需求，调整分片的数量
- **用量增加，更加去中心化**
  - 100 个席位 / 分片

信标链 (ETH & DOT) VS 夜影 (NEAR)

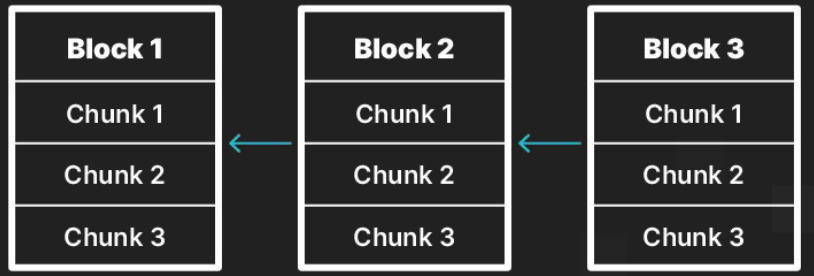
夜影协议可以处理每个区块内的分片，而不受限于信标链的扩容能力

Beacon Chain



Shard Chains

Nightshade



关于分片方法的更多细节，请参见:

英文：near.ai/nightshade　中文：公众号

易用性：开发者体验

1. 快速构建、测试、部署
    a. AssemblyScript & Rust (wasm)
    b. 完整的应用层工具
    c. 开发者无需考虑分片等复杂的底层理论
2. 隐藏区块链，为用户提供熟悉的体验
    a. 垫付gas费用
    b. 灵活的账户模型
3. 可持续的共享经济模式
    a. 30%手续费 -> 开发者
4. 可预测的交易费用

简洁的开发体验 -> 更快地构建、更频繁地迭代 -> 更快地找到客户

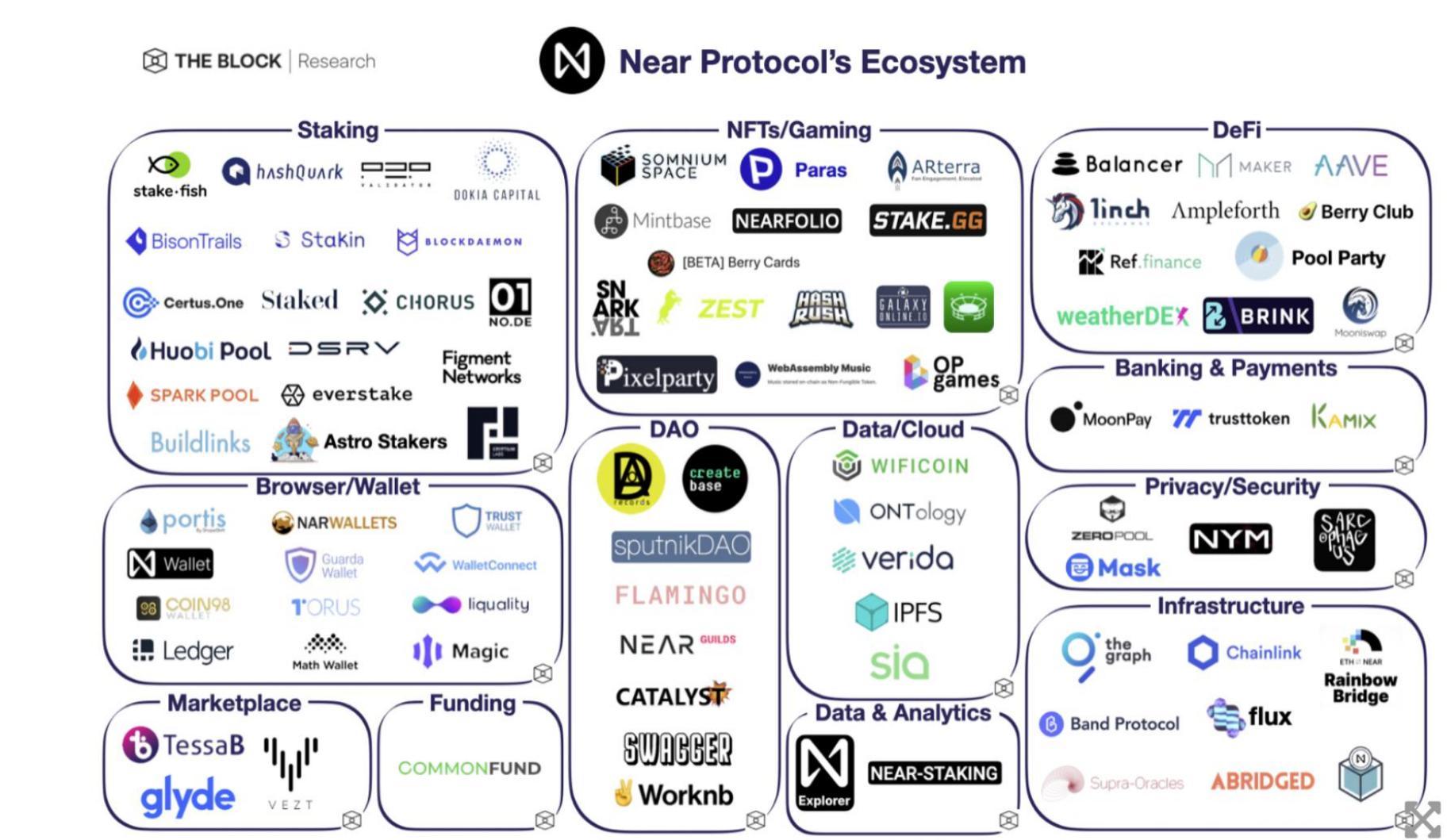
易用性：终端用户体验

1. 降低入门门槛
    a. 渐进式安全性
2. 开发者可以完全隐藏区块链
    a. 用户无需考虑交易（支付gas费）
    b. 无需了解公钥/私钥或备注词
3. 基于智能合约的可扩展账户模型
    a. 定制化的使用权限
    b. 简单可读的帐户名

在NEAR的平台上即使不懂区块链的用户也可以与链上应用无缝衔接，甚至没有通证也可以使用Dapp

THE BLOCK | Research

Near Protocol's Ecosystem



Scaling Ethereum is NEAR

aurora.dev

Aurora - EVM + 彩虹桥

Aurora 是最便捷的以太坊 L2 解决方案

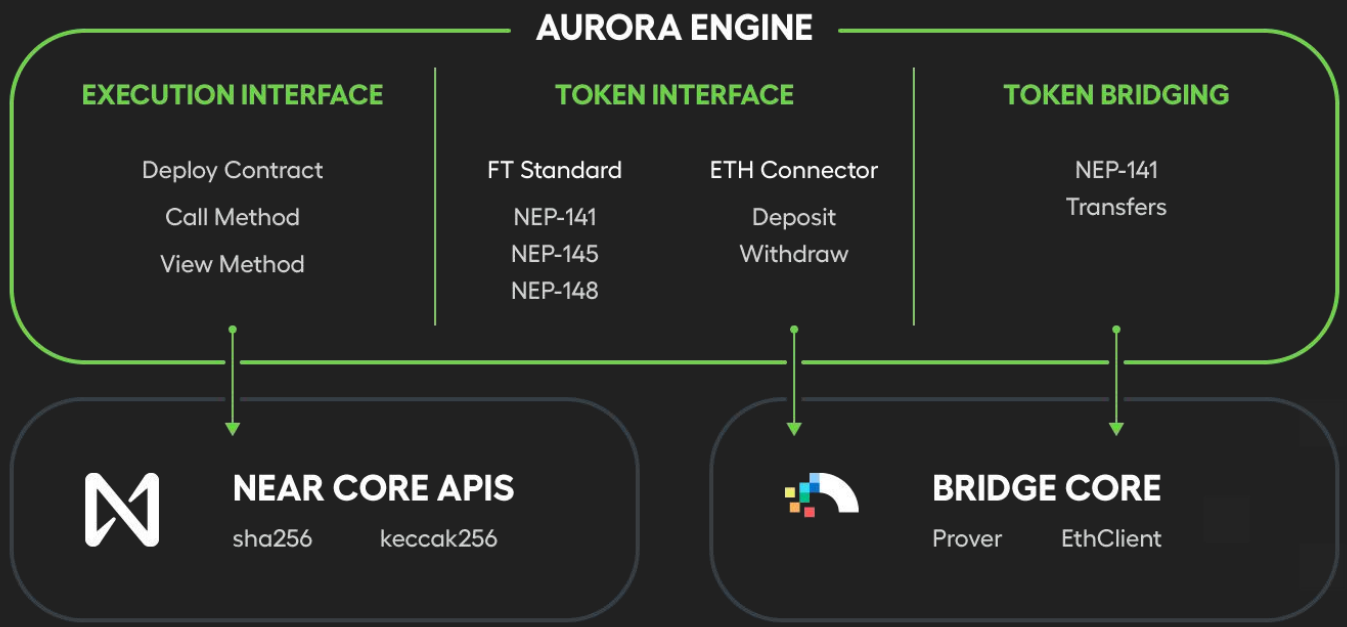
- Aurora 的核心是 SputnikVM — 来自 Parity Tech 的 Rust EVM 实现
- Aurora 支持以太坊现有的智能合约代码 — Solidity, Vyper
- Aurora 支持以太坊生态的现有的工具 — MetaMask, Truffle, HardHat, OpenZeppelin SDK 等
- **Aurora 使用 ETH 作为基础通证**
  - 交易费用以 ETH 计算
  - 开发者无需 fork 代码便可直接部署

Aurora - EVM + 彩虹桥

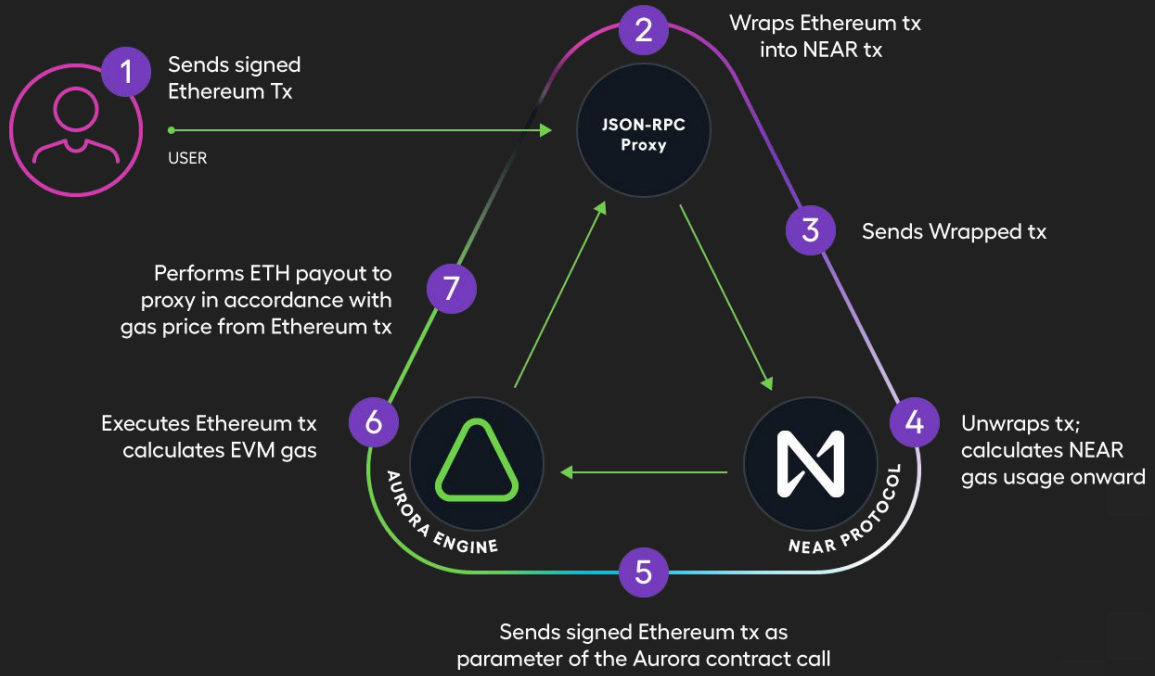
借助 NEAR 的优势

1. Gas费降低近千倍：平均交易执行成本低于0.01美元
2. 吞吐量：以太坊的50倍，可以通过动态分片算法实现横向扩容
3. 快速确认：执行交易仅需要1~2秒
4. 无需信任的桥：基于NEAR彩虹桥协议，支持ERC-20通证（目前已可用）、NFT、合约调用以及合约状态（仍在开发中）的互操作

Aurora Engine

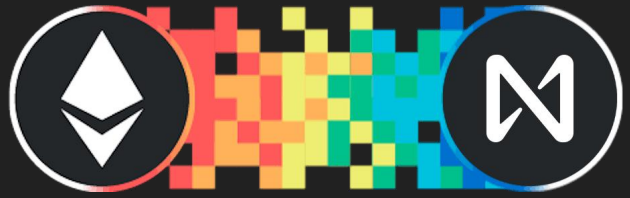


Aurora Engine



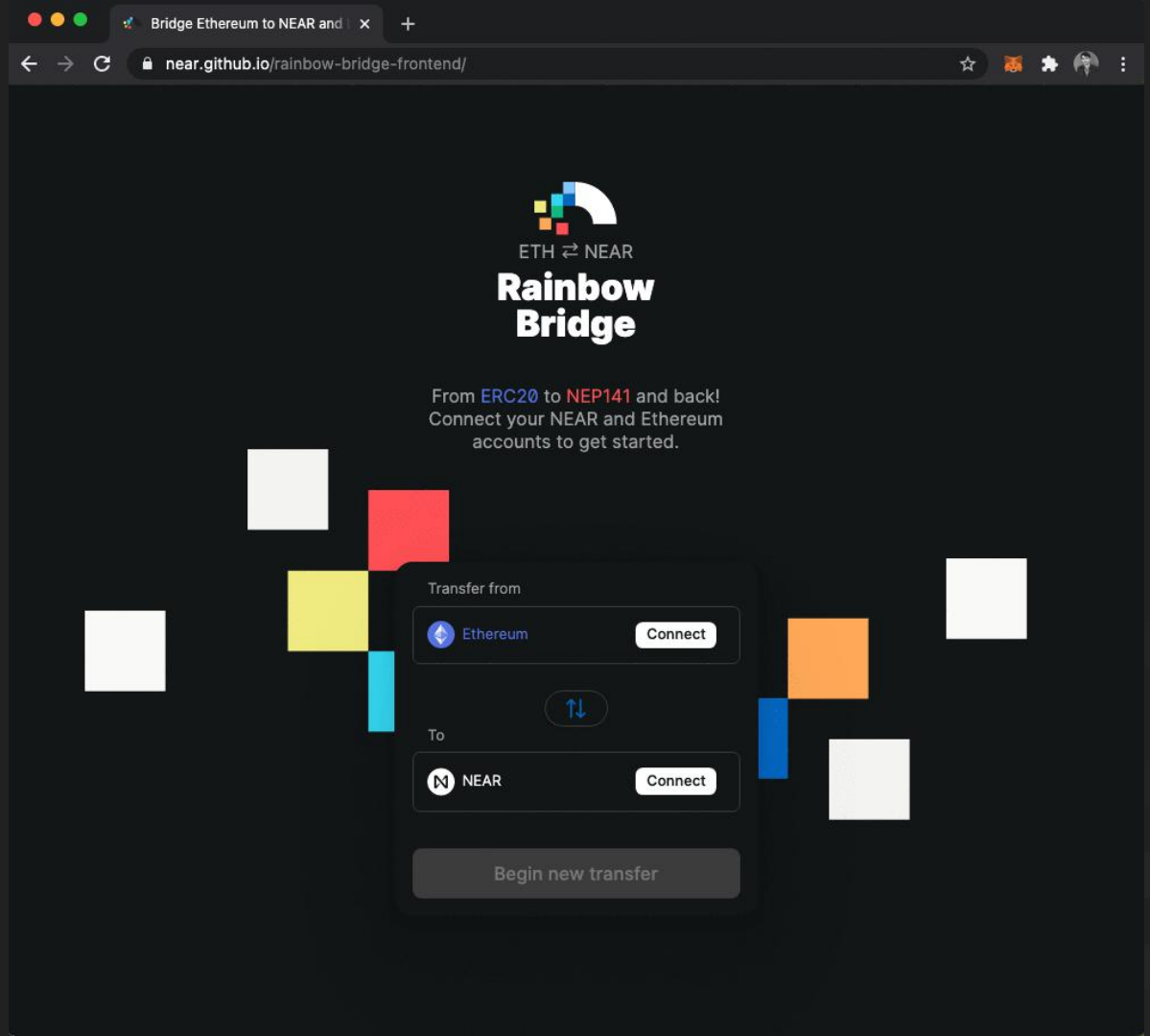
NEAR

彩虹桥



NEAR和以太坊之间的无需信任的桥

Demo

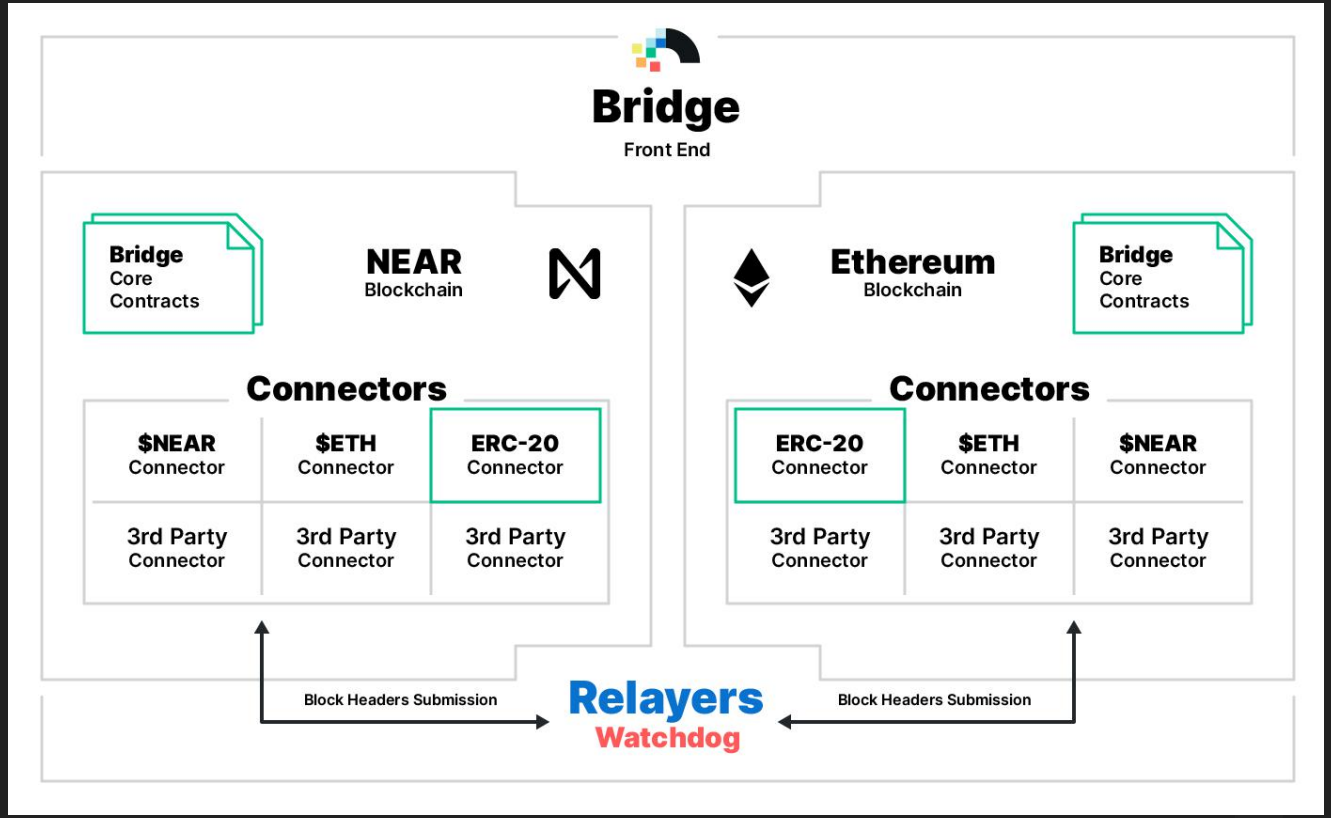

Bridge Ethereum to NEAR and
near.github.io/rainbow-bridge-frontend/
ETH ⇄ NEAR
Rainbow Bridge
From ERC20 to NEP141 and back!
Connect your NEAR and Ethereum accounts to get started.
Transfer from
Ethereum
Connect
To
NEAR
Connect
Begin new transfer

彩虹桥的特点

- 无需信任
  - 无需任何授权
  - 相信 >1/2 ETH 算力，>2/3 NEAR’s stake
- 去中心化
  - 任何人都可以维护/使用现有的桥
  - 或建立自己的桥梁
- 快速
  - ETH->NEAR = 7分钟，NEAR-> ETH= 10小时
  - 一旦EIP665实现，14秒内即可完成
- 通用
  - 不限于通证转账，支持多种互操作性
  - 如跨链合约调用

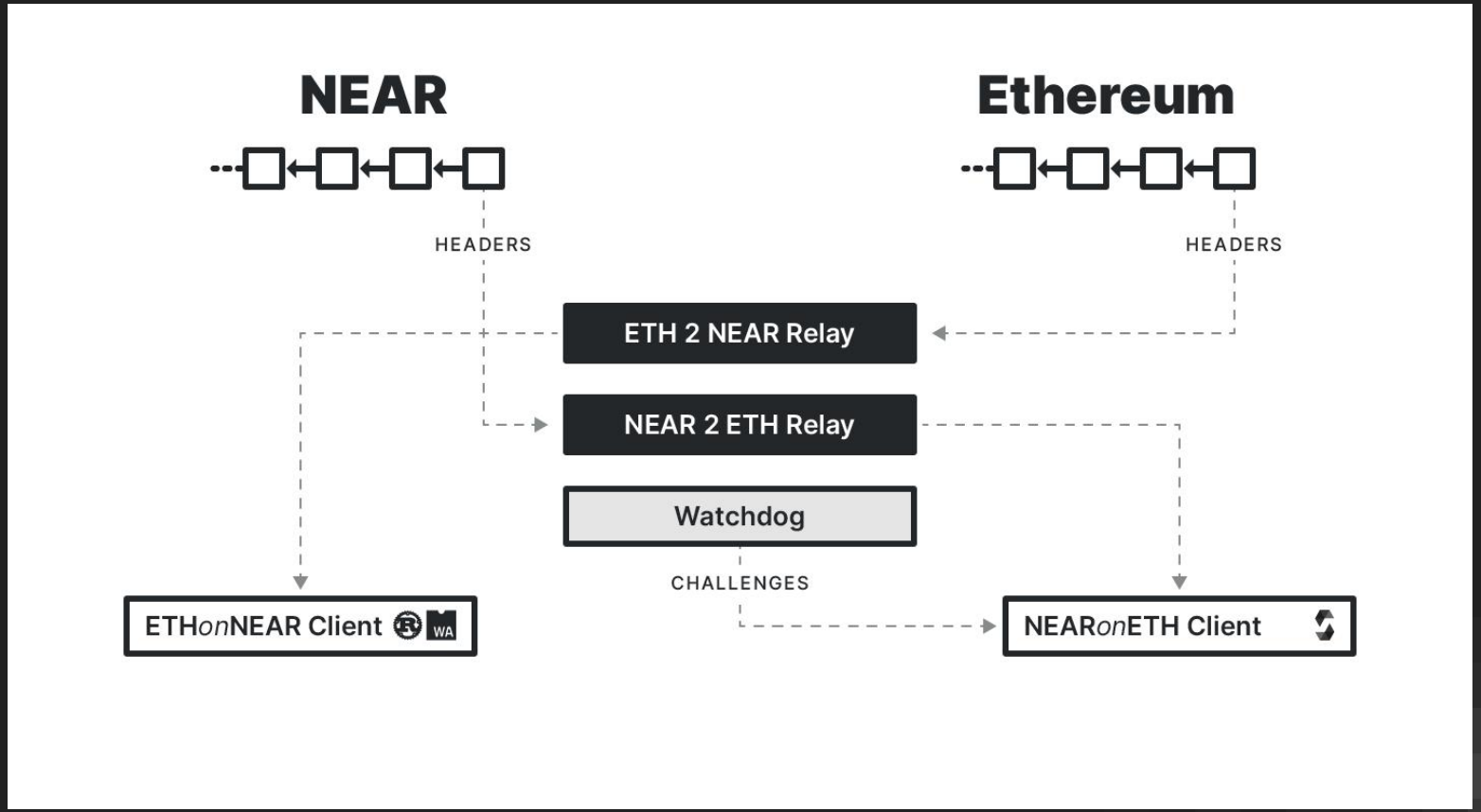
以太坊彩虹桥是互操作性的第一步。未来可利用相同架构桥接多条链，比如Polkadot、Cosmos

技术架构



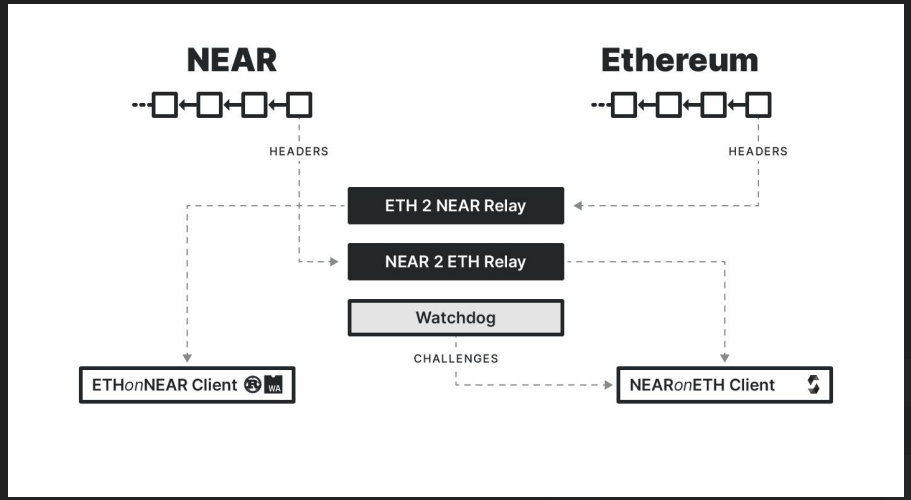
https://github.com/aurora-is-near/rainbow-bridge

技术架构


NEAR
Ethereum
HEADERS
HEADERS
ETH 2 NEAR Relay
NEAR 2 ETH Relay
Watchdog
CHALLENGES
ETHonNEAR Client
NEARonETH Client

Light Clients

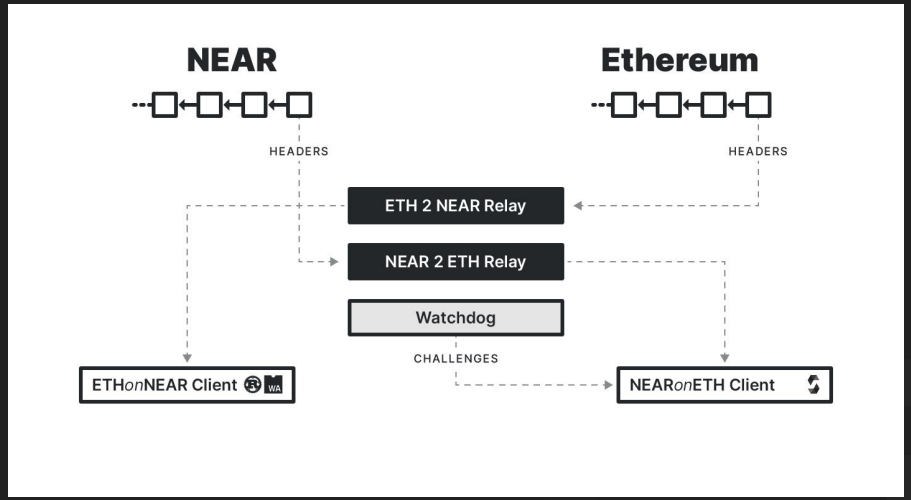
- EthOnNearClient
  - 检验 ETH headers
  - 用 Rust 实现的 NEAR 合约
- NearOnEthClient:
  - 检验 NEAR headers
  - 用 Solidity 实现的 Ethereum 合约



Relays

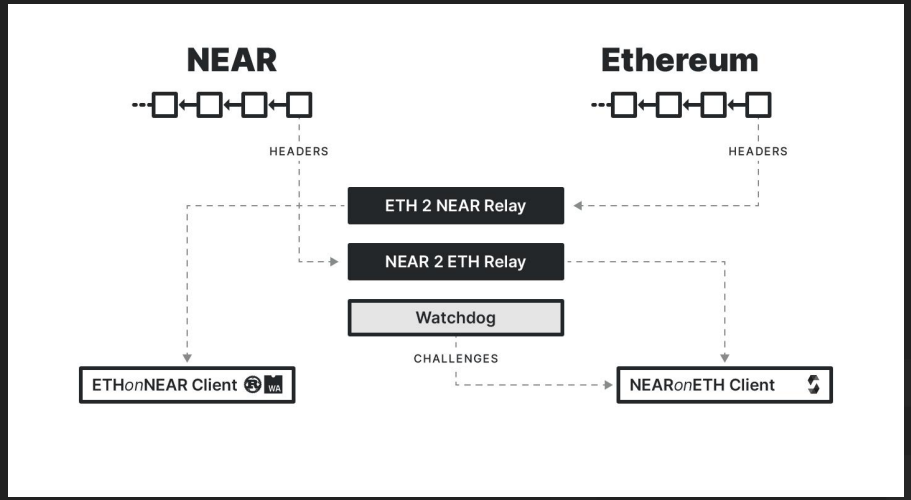
- Eth2NearRelay: 将每个 ETH header 发送给 EthOnNearClient 合约

- Near2EthRelay：每4小时将1个 NEAR header 发送给 NearOnEthClient 合约



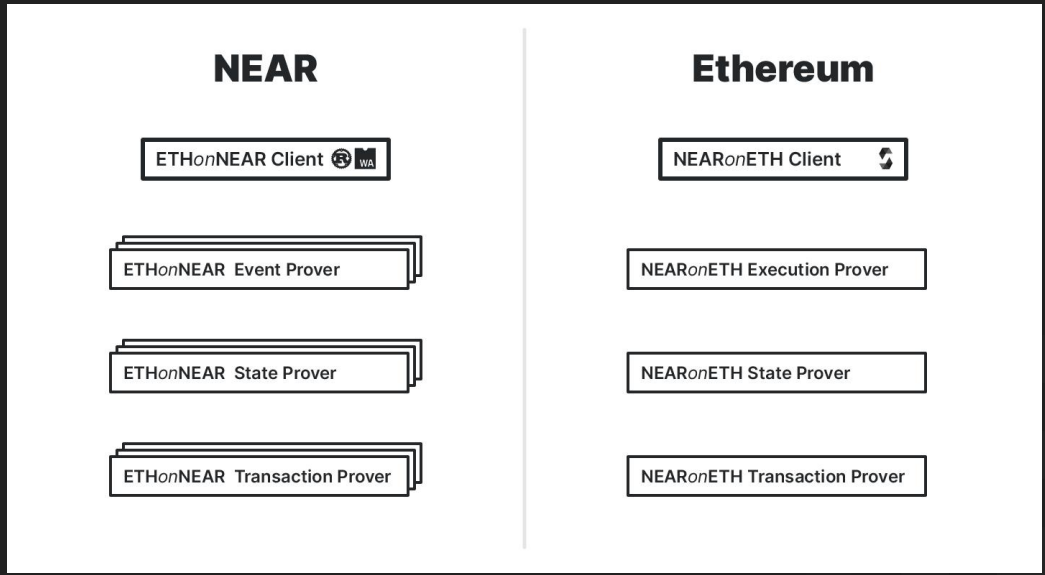
Watchdog

- 监控提交的 NEAR headers，挑战签名无效的 header
- 可运行多个 watchdog 服务以提高安全性
- 在 EIP665 被接收后，以太坊 EVM 将提供 Ed25519 签名的 precompile。那么可以取消 watchdog 服务和4小时的挑战窗口



Provers

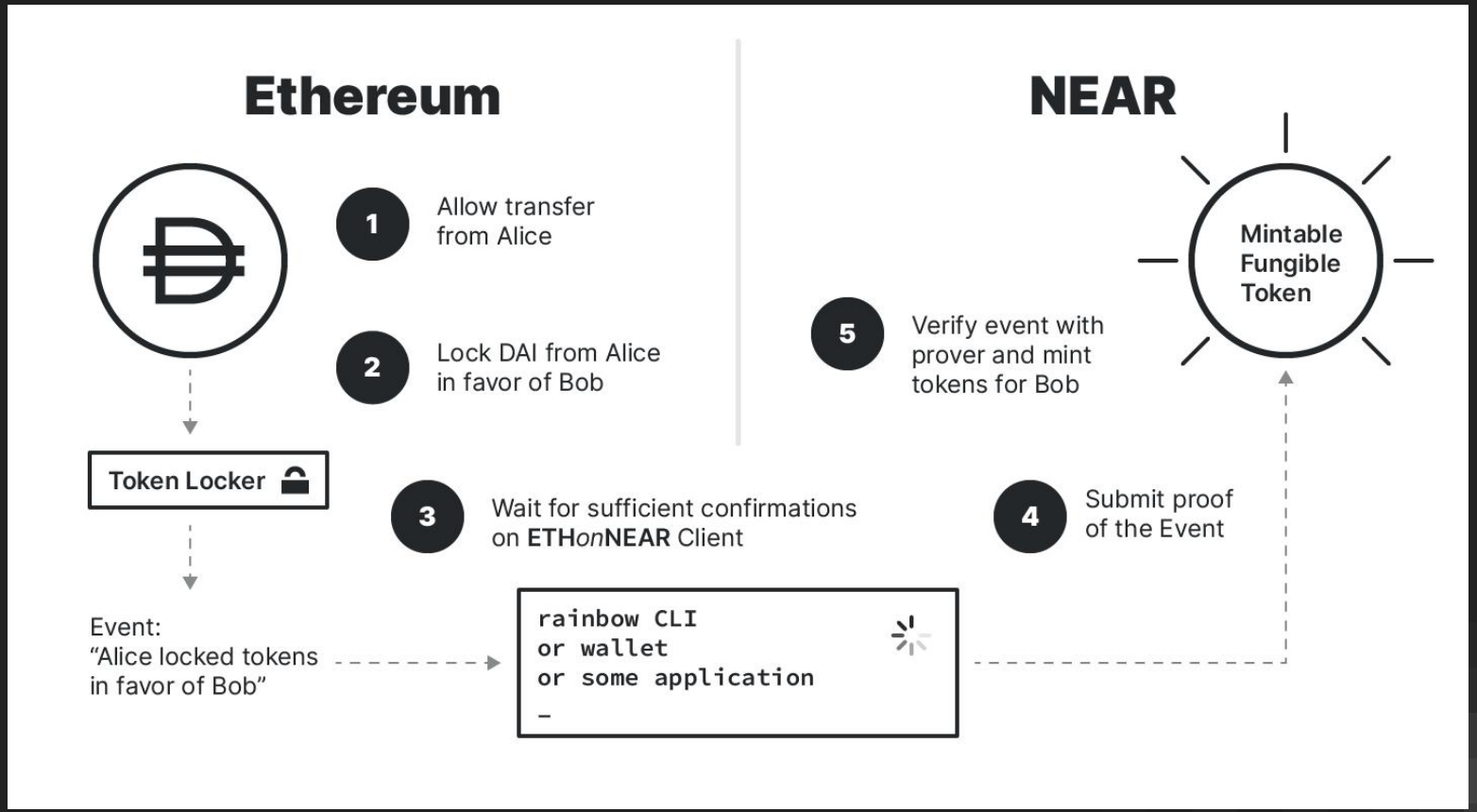
- Eth2NearProver
  - 验证 ETH 事件确实发生
- Near2EthProver
  - 验证 NEAR 执行结果

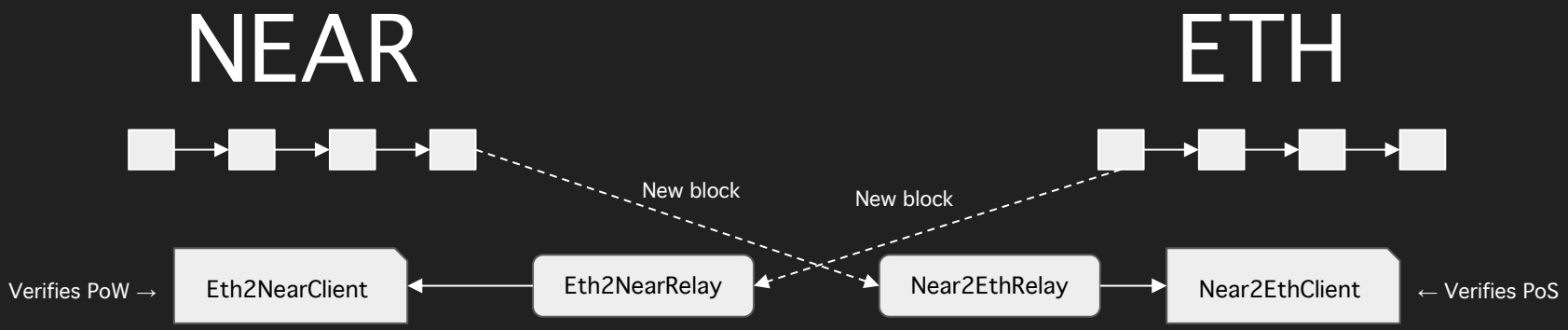


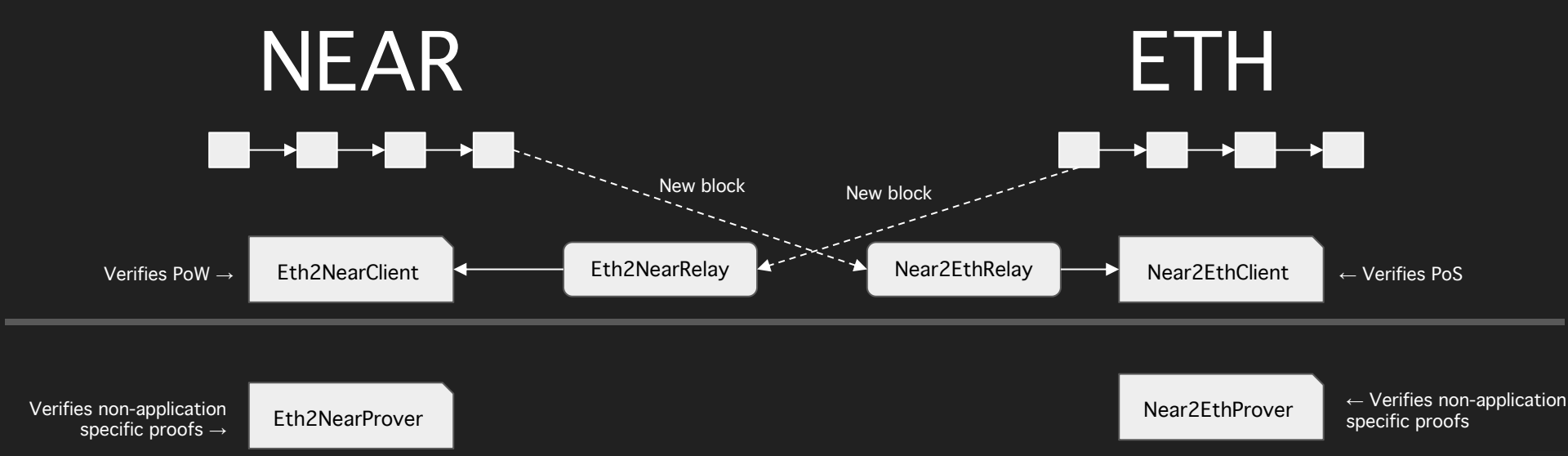
Connectors

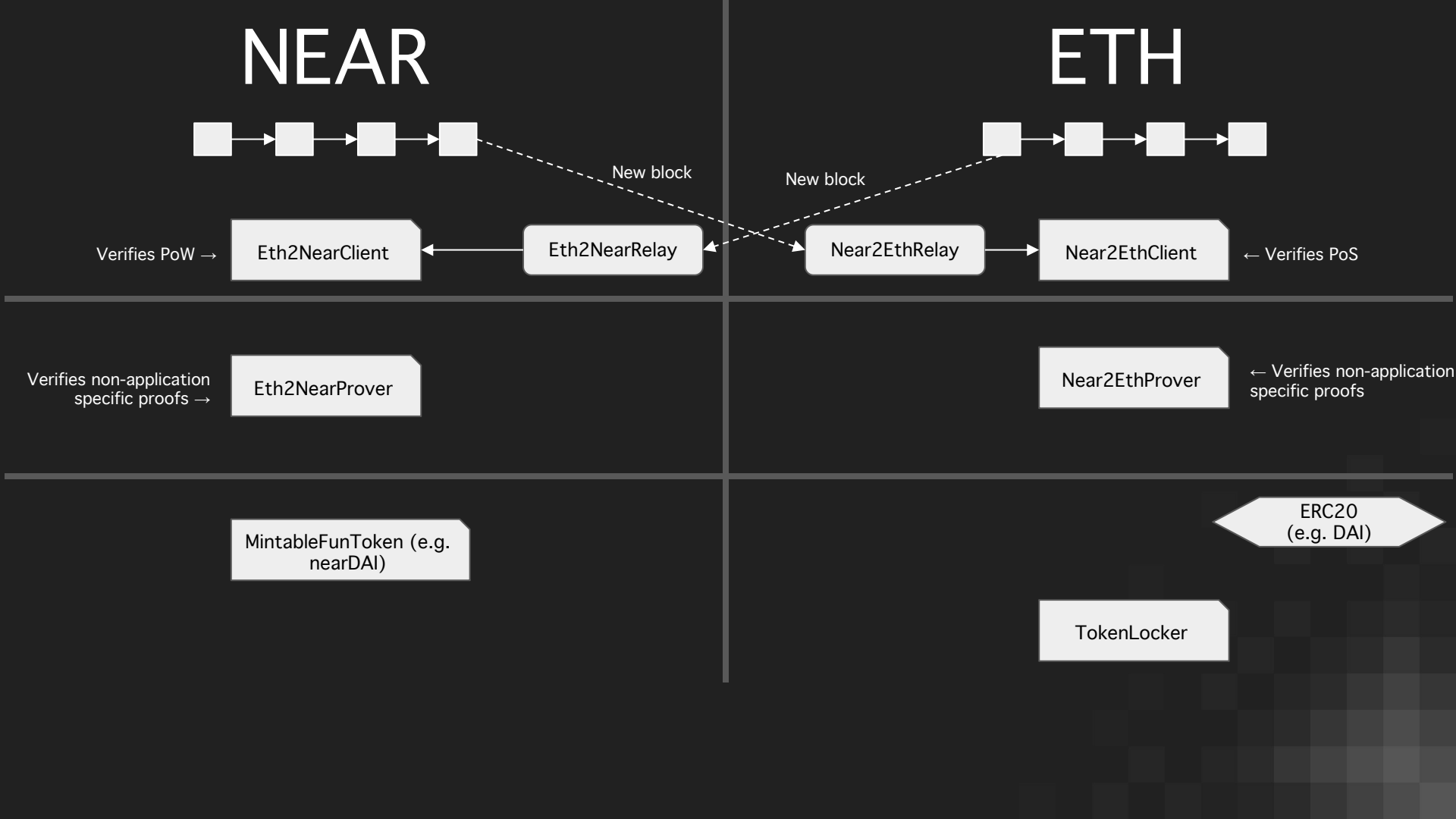
- 彩虹桥支持多种 connectors 扩展
- ERC20 Connector：通过合约上的 “locking”, “minting”, “burning” and “release” 等操作，实现 以太坊 ERC20 通证和 NEAR NEP141 通证之间的无需信任的资产转移。为了操作的稳定和安全，需要验证这些事件确实发生：如以太坊上的 DAI 确实被锁定了，如 NEAR 上的 nDAI 确实被销毁了

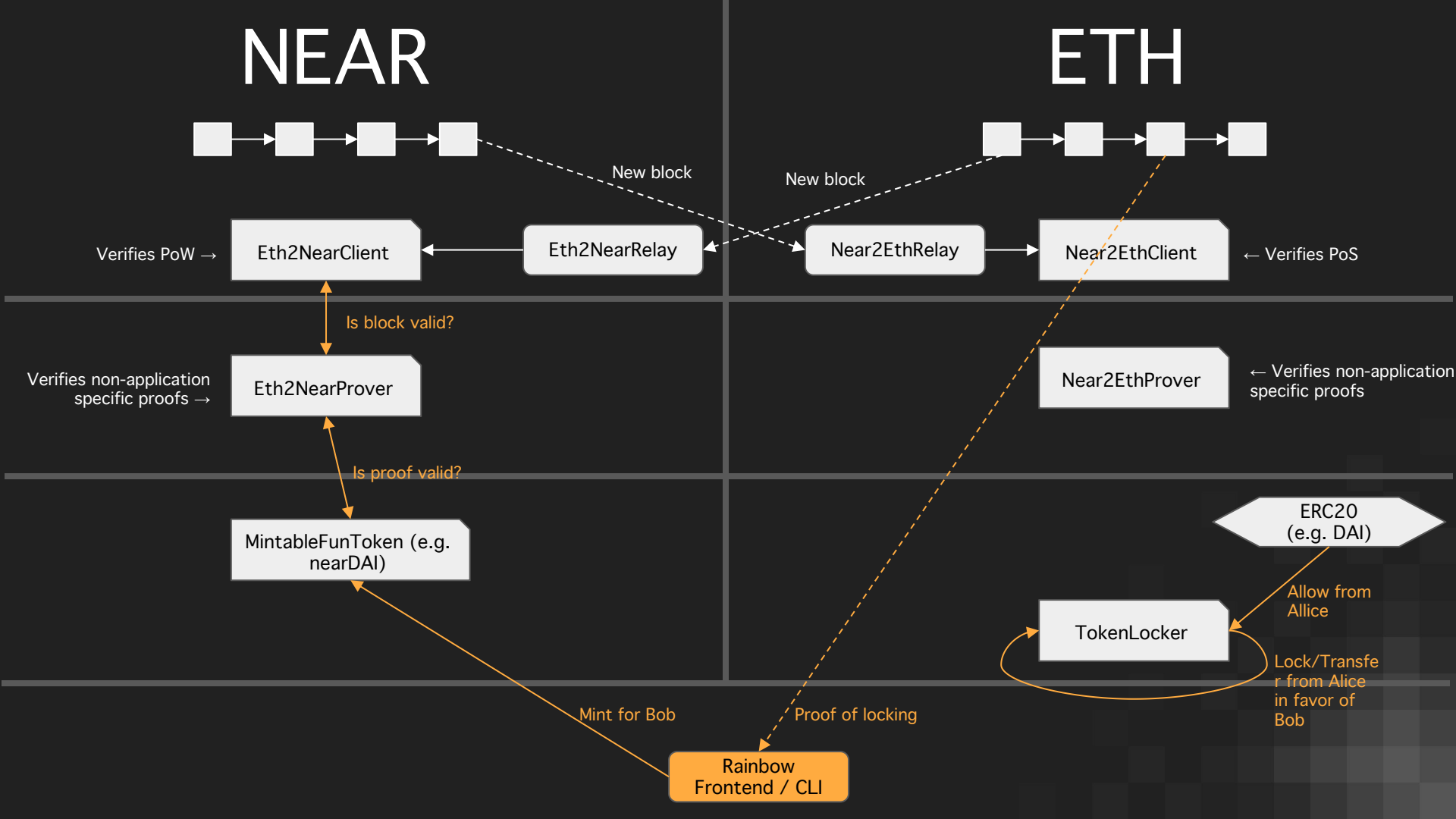
ERC20 Use Case




NEAR
ETH
New block
New block
Verifies PoW →
Eth2NearClient
Eth2NearRelay
Near2EthRelay
Near2EthClient
← Verifies PoS


NEAR
ETH
New block
New block
Verifies PoW →
Eth2NearClient
Eth2NearRelay
Near2EthRelay
Near2EthClient
← Verifies PoS
Verifies non-application specific proofs →
Eth2NearProver
Near2EthProver
← Verifies non-application specific proofs


NEAR
ETH
New block
New block
Verifies PoW →
Eth2NearClient
Eth2NearRelay
Near2EthRelay
Near2EthClient
← Verifies PoS
Verifies non-application specific proofs →
Eth2NearProver
Near2EthProver
← Verifies non-application specific proofs
MintableFunToken (e.g. nearDAI)
ERC20 (e.g. DAI)
TokenLocker


NEAR
ETH
New block
New block
Verifies PoW →
Eth2NearClient
Eth2NearRelay
Near2EthRelay
Near2EthClient
← Verifies PoS
Is block valid?
Verifies non-application specific proofs →
Eth2NearProver
Near2EthProver
← Verifies non-application specific proofs
Is proof valid?
MintableFunToken (e.g. nearDAI)
ERC20 (e.g. DAI)
Allow from Allice
TokenLocker
Lock/Transfer from Alice in favor of Bob
Mint for Bob
Proof of locking
Rainbow Frontend / CLI

TokenLocker

```
function lockToken(address ethToken, uint256 amount, string memory accountId)
    public
    pausable (PAUSED_LOCK)
{
    require(IERC20(ethToken).balanceOf(address(this)).add(amount) <= ((uint256(1) << 128) - 1), "Maximum tokens locked exceeded (< 2^128 - 1)");
    IERC20(ethToken).safeTransferFrom(msg.sender, address(this), amount);
    emit Locked(address(ethToken), msg.sender, amount, accountId);
}

function unlockToken(bytes memory proofData, uint64 proofBlockHeight)
    public
    pausable (PAUSED_UNLOCK)
{
    ProofDecoder.ExecutionStatus memory status = _parseAndConsumeProof(proofData, proofBlockHeight);
    BurnResult memory result = _decodeBurnResult(status.successValue);
    IERC20(result.token).safeTransfer(result.recipient, result.amount);
    emit Unlocked(result.amount, result.recipient);
}
```

https://github.com/aurora-is-near/rainbow-token-connector/blob/23b0c3aca6c5755b70cb6117d20c12e9a640fc37/erc20-connector/contracts/ERC20Locker.sol#L51-L68

BridgeToken

```rust
#[payable]
pub fn mint(&mut self, account_id: AccountId, amount: U128) {
    assert_eq!(
        env::predecessor_account_id(),
        self.controller,
        "Only controller can call mint"
    );

    self.storage_deposit(Some(account_id.as_str().try_into().unwrap()), None);
    self.token.internal_deposit(&account_id, amount.into());
}

#[payable]
pub fn withdraw(&mut self, amount: U128, recipient: String) -> Promise {
    self.check_not_paused(PAUSE_WITHDRAW);

    assert_one_yocto();
    Promise::new(env::predecessor_account_id()).transfer(1);

    self.token
        .internal_withdraw(&env::predecessor_account_id(), amount.into());

    ext_bridge_token_factory::finish_withdraw(
        amount.into(),
        recipient,
        &self.controller,
        NO_DEPOSIT,
        FINISH_WITHDRAW_GAS,
    )
}
```

https://github.com/aurora-is-near/rainbow-token-connector/blob/a45e23aed6d3566ea8f89da7d9527da931699fe6/bridge-token/src/lib.rs#L80-L109

加入我们

共同打造开放网络的未来！

开发者:
near.chat
微信群

创业者:
openwebcollective.com

社区:
near.chat
t.me/cryptonear