

一、系统介绍

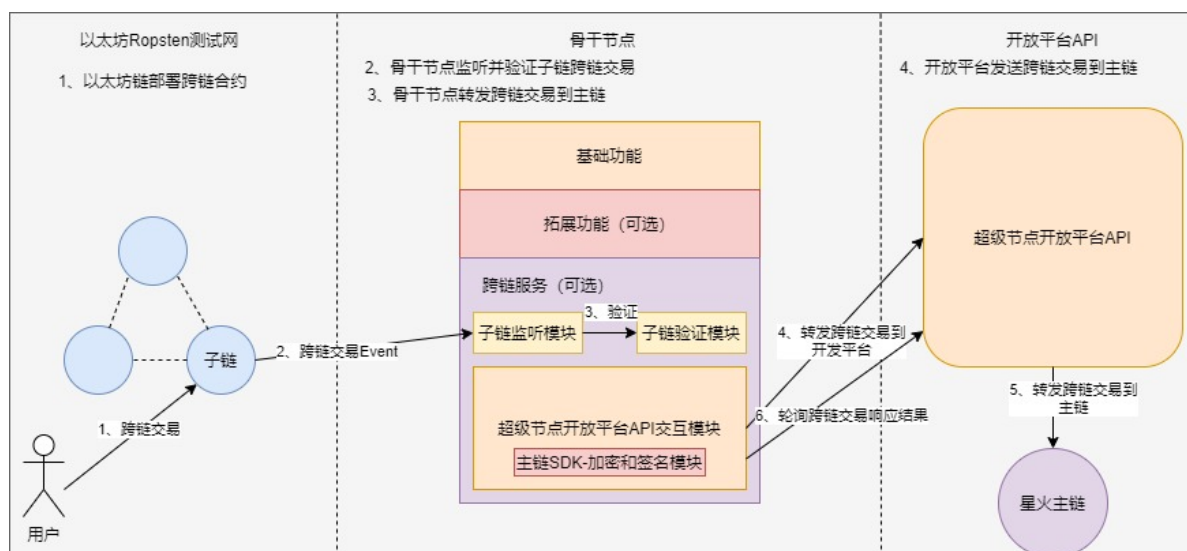
本系统通过搭建骨干节点网关服务，利用“公证人”机制，为跨链交易进行转发和验证，以实现星火主链与以太坊链进行跨链互操作。骨干节点网关服务主要包括三大模块：跨链事件监听模块、跨链事件验证模块和跨链事件转发模块。

跨链事件监听模块：以太坊链按要求开发跨链智能合约，用户通过调用合约触发跨链事件，骨干节点使用以太坊的SDK来时刻监听到这些跨链事件。对于星火主链上发起的跨链交易，骨干节点则通过超级节点开发平台API提供的跨链服务，来轮询获取主链上的跨链交易，完成监听的功能。

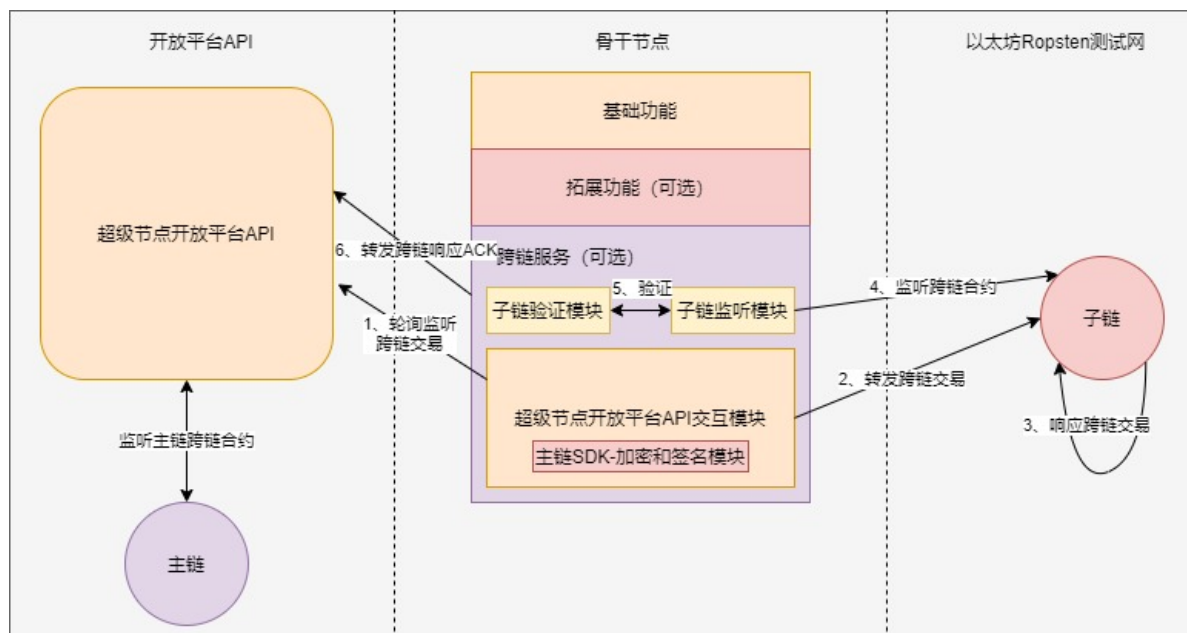
跨链事件验证模块：骨干节点需要对子链发起的跨链交易进行验证，并将验证结果和跨链交易一起提交到主链上，主链对验证证明进行备份存储，可以采用抽验的方式进行二次验证，也能够为后续的问责机制提供数字证明。

跨链事件转发模块：以太坊链和星火主链之间的跨链交易，需要骨干节点进行转发。骨干节点通过以太坊的跨链智能合约来获取跨链交易详情，针对不同类型的跨链交易进行处理，进而有选择的调用超级节点开放平台API提供的接口服务，将跨链交易转发到星火主链上。反之一样，骨干节点通过超级节点开放平台API提供的接口服务，来获取主链上的跨链交易详情，然后对交易进行处理组装，最后转发到以太坊链上。

跨链交易子链到主链的流程：



跨链交易主链到子链的流程：



二、部署使用

1.环境准备

环境准备是部署和使用以太坊跨链demo的第一步，主要是说明demo及相关组件运行的硬件配置和软件依赖，您需要在部署跨链demo之前确认机器满足下述的要求。

1.1 硬件

配置	推荐配置	最低配置
CPU	2.4GHz	1.5GHz
内存	16GB	8GB
核心	8核	4核
带宽	10Mb	2Mb

1.2 系统

- Windows 7 或者 Windows 10
- Ubuntu 16.04 及以上
- CentOS 7.2 及以上

1.3 软件

- 星火·链网超级节点开放平台API
- Nodejs
- Postman
- MetaMask

1.4 文件结构

```
├─build                                //truffle合约编译
|   └─contracts
├─contracts                            //solidity跨链合约,主要为CrossChainContract.sol和
CrossChainContractAdditional.sol
├─demo                                //骨干网关节点跨链服务,lib为主链签名SDK,node_modules
包括web3js、request、ethereumjs-tx等
|   └─lib                              //backbone_bif.js和backbone_eth.js为与以太坊私有链
交互的骨干网关服务:
|       └─bifchain-js-signer          //backbone_server_bif.js和
backbone_server_eth.js为与以太坊测试网交互的骨干网关服务;
|           └─node_modules
├─images
├─lib                                  //库合约
├─migrations                          //truffle部署跨链合约
└─test
```

2.部署说明

2.1 以太坊链环境搭建

2.1.1 Ropsten测试网

Ropsten是以太坊官方提供的测试网络，是为了解决Morden难度炸弹问题而重新启动的一条区块链，目前仍在运行，共识机制为PoW。本跨链服务既可以在本地以太坊私有链环境下使用，也可以子在测试网环境运行，本文主要讲与测试网环境进行跨链交互。

2.1.2 跨链合约编译

以太坊的跨链合约主要为CrossChainContract.sol和CrossChainContractAdditional.sol（补充合约），可以使用remix进行编译，编译器版本建议选择0.7.0，同时由于跨链合约过大需要使用优化编译Enable optimization，得到合约的ABI。


```

[{"anonymous": false, "inputs":
[{"indexed": false, "internalType": "string", "name": "operation", "type": "string"},
{"indexed": false, "internalType": "address", "name": "from", "type": "address"},
{"indexed": false, "internalType": "uint256", "name": "value", "type": "uint256"}], "name":
"AssetBurnEvent", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "internalType": "string", "name": "operation", "type": "string"},
{"indexed": false, "internalType": "address", "name": "from", "type": "address"},
{"indexed": false, "internalType": "address", "name": "to", "type": "address"},
{"indexed": false, "internalType": "uint256", "name": "value", "type": "uint256"}], "name":
"AssetMintEvent", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "internalType": "string", "name": "operation", "type": "string"},
{"indexed": false, "internalType": "uint256", "name": "value", "type": "uint256"}], "name":
"approveEvent", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "internalType": "string", "name": "operation", "type": "string"},
{"indexed": false, "internalType": "string", "name": "cross_TxNo", "type": "string"},
{"indexed": false, "internalType": "uint8", "name": "txType", "type": "uint8"}], "name":
"sendAckedEvent", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "internalType": "string", "name": "operation", "type": "string"},
{"indexed": false, "internalType": "string", "name": "cross_TxNo", "type": "string"},
{"indexed": false, "internalType": "uint8", "name": "txType", "type": "uint8"}], "name":
"sendTxEvent", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "internalType": "string", "name": "operation", "type": "string"},
{"indexed": false, "internalType": "string", "name": "chain_Code", "type": "string"}], "name":
"setChainCodeEvent", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "internalType": "string", "name": "operation", "type": "string"},
{"indexed": false, "internalType": "address", "name": "gateway_Address", "type": "address"}], "name":
"setGatewayEvent", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "internalType": "string", "name": "operation", "type": "string"},
{"indexed": false, "internalType": "string", "name": "cross_TxNo", "type": "string"},
{"indexed": false, "internalType": "uint8", "name": "txType", "type": "uint8"}], "name":
"startTxEvent", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "internalType": "string", "name": "operation", "type": "string"},
{"indexed": false, "internalType": "address", "name": "to", "type": "address"}], "name":
"takeOutEvent", "type": "event"}, {"anonymous": false, "inputs":
[{"indexed": false, "internalType": "string", "name": "operation", "type": "string"},
{"indexed": false, "internalType": "uint256", "name": "value", "type": "uint256"}], "name":
"transferEvent", "type": "event"}, {"inputs":
[{"internalType": "address", "name": "from", "type": "address"},
{"internalType": "address", "name": "to", "type": "address"},
{"internalType": "uint256", "name": "value", "type": "uint256"}], "name": "approve", "outputs": [], "stateMutability": "nonpayable", "type": "function"}, {"inputs":
[{"internalType": "address", "name": "addr", "type": "address"}], "name": "balanceOf", "outputs":
[{"internalType": "uint256", "name": "", "type": "uint256"}], "stateMutability": "view", "type": "function"}, {"inputs":
[{"internalType": "address", "name": "owner", "type": "address"},
{"internalType": "address", "name": "spender", "type": "address"}], "name": "getAllowance", "outputs":
[{"internalType": "uint256", "name": "", "type": "uint256"}], "stateMutability": "view", "type": "function"}, {"inputs":
[{"internalType": "string", "name": "crossTxNo", "type": "string"}], "name": "getCrossTx", "outputs":
[{"internalType": "bytes", "name": "", "type": "bytes"}, {"internalType": "bytes", "name": "", "type": "bytes"}, {"internalType": "enum Storage.TxResultEnum", "name": "", "type": "uint8"}, {"internalType": "enum Storage.TxRefundedEnum", "name": "", "type": "uint8"}, {"internalType": "enum Storage.TxOriginEnum", "name": "", "type": "uint8"}, {"internalType": "bytes", "name": "", "type": "bytes"}, {"internalType": "bytes", "name": "", "type": "bytes"}],

```

```

{"internalType":"bytes","name":"","type":"bytes"}], "stateMutability":"view", "type":
"function"}, {"inputs": [], "name": "getVersion", "outputs":
[{"internalType":"string","name":"","type":"string"}], "stateMutability":"view", "
type":"function"}, {"inputs":
[{"internalType":"address","name":"crossChainAdd","type":"address"}], "name":"ini
tialize", "outputs": [], "stateMutability":"nonpayable", "type":"function"},
{"inputs": [{"internalType":"string","name":"crossTxNo","type":"string"},
{"internalType":"enum Storage.TxResultEnum","name":"result","type":"uint8"},
{"internalType":"string","name":"version","type":"string"},
{"internalType":"bytes","name":"proof","type":"bytes"}], "name":"sendAcked", "outp
uts": [], "stateMutability":"nonpayable", "type":"function"}, {"inputs":
[{"internalType":"string","name":"srcBid","type":"string"},
{"internalType":"address payable","name":"destAddress","type":"address"},
{"internalType":"string","name":"srcChainCode","type":"string"},
{"internalType":"string","name":"destChainCode","type":"string"},
{"internalType":"uint8","name":"txType","type":"uint8"},
{"internalType":"string","name":"crossTxNo","type":"string"},
{"internalType":"bytes","name":"payload","type":"bytes"},
{"internalType":"string","name":"extension","type":"string"},
{"internalType":"string","name":"remark","type":"string"},
{"internalType":"string","name":"version","type":"string"},
{"internalType":"bytes","name":"proof","type":"bytes"}], "name":"sendTx", "outputs
": [], "stateMutability":"payable", "type":"function"}, {"inputs":
[{"internalType":"string","name":"chainCode","type":"string"}], "name":"setChainC
ode", "outputs": [], "stateMutability":"nonpayable", "type":"function"}, {"inputs":
[{"internalType":"address","name":"gatewayAddress","type":"address"}], "name":"se
tGateway", "outputs": [], "stateMutability":"nonpayable", "type":"function"},
{"inputs": [{"internalType":"address","name":"srcAddress","type":"address"},
{"internalType":"string","name":"destBid","type":"string"},
{"internalType":"string","name":"srcChainCode","type":"string"},
{"internalType":"string","name":"destChainCode","type":"string"},
{"internalType":"uint8","name":"txType","type":"uint8"},
{"internalType":"bytes","name":"payload","type":"bytes"},
{"internalType":"string","name":"extension","type":"string"},
{"internalType":"string","name":"remark","type":"string"},
{"internalType":"string","name":"version","type":"string"}], "name":"startTx", "ou
tputs": [], "stateMutability":"payable", "type":"function"}, {"inputs":
[{"internalType":"string","name":"crossTxNo","type":"string"},
{"internalType":"address payable","name":"to","type":"address"}], "name":"takeOut", "outputs":
[], "stateMutability":"nonpayable", "type":"function"}, {"inputs":
[{"internalType":"address","name":"to","type":"address"},
{"internalType":"uint256","name":"value","type":"uint256"}], "name":"transfer", "o
utputs": [], "stateMutability":"nonpayable", "type":"function"}]

```

2.1.3 部署跨链合约

合约的部署采用remix，以太坊跨链合约已部署到Ropsten测试网，合约地址：
0x717649243802CB60c347Eb76622B000F1Dd74F30。

2.1.4 跨链合约初始化

```

//合约初始化只能进行一次
const crossAdditional = "0x970994c0c74870b60feB9A93F862EeF1DC8DB8a2"
const contractInit = myContract.methods.initialize(crossAdditional)
const contractInitAbi = contractInit.encodeABI();

```

```

let estimatedGas;
let nonce;

contractInit.estimateGas({from: deployerAccount}).then((gasAmount) => {
    estimatedGas = gasAmount.toString(16);

    console.log("Estimated gas: " + estimatedGas);

    web3.eth.getTransactionCount(deployerAccount).then(_nonce => {
        nonce = _nonce.toString(16);

        console.log("Nonce: " + nonce);
        const txParams = {
            gasPrice: 10000000000,
            gasLimit: 3000000,
            to: crossChainContractAddress,
            data: contractInitAbi,
            from: deployerAccount,
            nonce: '0x' + nonce
        };

        const tx = new Tx(txParams, {chain: 'ropsten', hardfork: 'petersburg'});
        tx.sign(deployerPrivateKey);

        const serializedTx = tx.serialize();

        web3.eth.sendSignedTransaction('0x' +
        serializedTx.toString('hex')).on('receipt', receipt => {
            console.log(receipt);
        });
    });
});

```

2.1.5 设置以太坊chainCode

```

//设置的chainCode必须是向主链申请注册通过的,具体方法请参考星火·链网生态接入开发规范
const _ChainCode = "a202"
const contractSetChainCode = myContract.methods.setChainCode(_ChainCode)
const contractSetChainCodeAbi = contractSetChainCode.encodeABI();

let estimatedGas;
let nonce;

contractSetChainCode.estimateGas({from: deployerAccount}).then((gasAmount) => {
    estimatedGas = gasAmount.toString(16);

    console.log("Estimated gas: " + estimatedGas);

    web3.eth.getTransactionCount(deployerAccount).then(_nonce => {
        nonce = _nonce.toString(16);

        console.log("Nonce: " + nonce);
        const txParams = {
            gasPrice: 10000000000,
            gasLimit: 3000000,
            to: crossChainContractAddress,
            data: contractSetChainCodeAbi,

```

```

    from: deployerAccount,
    nonce: '0x' + nonce
  });

  const tx = new Tx(txParams, {chain: 'ropsten', hardfork: 'petersburg'});
  tx.sign(deployerPrivateKey);

  const serializedTx = tx.serialize();

  web3.eth.sendSignedTransaction('0x' +
    serializedTx.toString('hex')).on('receipt', receipt => {
    console.log(receipt);
  })
  });
});

```

2.1.6 设置以太坊网关节点

```

//负责监听以太坊的跨链合约事件，以及转发和反馈跨链事件
const gatewayAddress = "0xE1fAdFBba60313C932Fe538a400aDf37d6308f0B"
//const gatewayPrivateKey =
"9e5e9bc3c721539c949b10cec8ce2d6079cf181d5ccfbd78c81e052ee6f62394"
const contractSetGateway = myContract.methods.setGateway(gatewayAddress)
const contractSetGatewayAbi = contractSetGateway.encodeABI();

let estimatedGas;
let nonce;

console.log("Getting gas estimate");

contractSetGateway.estimateGas({from: deployerAccount}).then((gasAmount) => {
  estimatedGas = gasAmount.toString(16);

  console.log("Estimated gas: " + estimatedGas);

  web3.eth.getTransactionCount(deployerAccount).then(_nonce => {
    nonce = _nonce.toString(16);

    console.log("Nonce: " + nonce);
    const txParams = {
      gasPrice: 10000000000,
      gasLimit: 3000000,
      to: crossChainContractAddress,
      data: contractSetGatewayAbi,
      from: deployerAccount,
      nonce: '0x' + nonce
    };

    const tx = new Tx(txParams, {chain: 'ropsten', hardfork: 'petersburg'});
    tx.sign(deployerPrivateKey);

    const serializedTx = tx.serialize();

    web3.eth.sendSignedTransaction('0x' +
      serializedTx.toString('hex')).on('receipt', receipt => {
        console.log(receipt);
      })
  });
});

```



```
    })  
  });  
});
```

2.2 星火主链环境搭建

参考星火·链网超级节点开放平台API文档

2.3 骨干网关节点搭建

网关节点主要负责跨链交易的监听，转发和验证。网关服务由backbone_server_bif.js和backbone_server_eth.js两部分组成，分别负责监听和转发星火主链和以太坊测试网上的跨链交易。网关与以太坊的交互使用web3js，与星火主链交互使用http。

2.3.1 与以太坊链交互

监听并转发以太坊跨链合约事件，每5秒一次，可以自行调整。

```
var rule = new schedule.RecurrenceRule();  
var times = [];  
for(var i=1; i<60; ){  
    //每5秒查询一次跨链事件  
    i += 5;  
    times.push(i);  
}  
rule.second = times;  
  
schedule.scheduleJob(rule, async() => {  
    //获取当前区块号  
    var blockHeight = await web3.eth.getBlockNumber();  
    console.log("schedule", "blockHeight:", blockHeight);  
    if (blockNum < blockHeight) {  
        blockNum = blockHeight  
        await getEvents(blockHeight, blockHeight);  
    }  
})  
  
let getEvents = async (fromBlock, toBlock) =>{  
  
    console.log('fromBlock:', fromBlock, 'toBlock:', toBlock);  
    //监听3种跨链事件  
    myContract.getPastEvents('allEvents', {  
        filter: {operation: ['startTxEvent', 'sendTxEvent', 'sendAckedEvent']},  
        fromBlock: fromBlock,  
        toBlock: toBlock,  
    },  
    async (error, events) => {  
        for (const event of events){  
  
            //获取跨链事件的接口类型、跨链编号、跨链类型  
            var _carossTxState = event.returnValues[0]  
            var _crossTxNo = event.returnValues[1]  
            var _crossTxType = event.returnValues[2]  
  
            //获取区块号、交易的哈希  
            var _ledgerSeq = event.blockNumber  
            var _txHash = event.transactionHash
```

```

        if (_carossTxState == 'startTx')
        {
            //用户发送跨链交易事件，网关节点转发跨链交易
            _startTx(_crossTxNo, _crossTxType, _ledgerSeq, _txHash)
        }
        else if (_carossTxState == 'sendTx')
        {
            //网关节点转发的跨链交易成功上链，需要回Ack给源链
            _sendTx(_crossTxNo, _ledgerSeq, _txHash)
        }
        else if (_carossTxState == 'sendAcked')
        {
            //网关节点回的Ack上链成功，需要给目标链回复Ack
            _sendAcked(_crossTxNo, _ledgerSeq, _txHash)
        }
    }
}
})
}

```

2.3.2 与星火主链交互

监听并处理转发星火主链跨链合约事件，每5秒一次，可以自行调整。

```

var rule = new schedule.RecurrenceRule();
var times = [];
for(var i=1; i<60; ){
    //每5秒查询一次跨链事件
    i += 5;
    times.push(i);
}
rule.second = times;

schedule.scheduleJob(rule, async() => {
    console.log("schedule", "pageStart:", pageStart);
    await getEvents(listURL, queryListParams, pageStart);
})

//获取主链跨链合约事件列表
let getEvents = async (_listURL, _queryListParams, _pageStart) =>{
    _queryListParams.params.pageStart = _pageStart
    request({
        url: _listURL,
        method: "POST",
        json: true,
        headers: "",
        body: _queryListParams
    }, function (error, response, body) {
        if (!error && response.statusCode == 200) {
            //记录总的跨链交易数量
            txTotal = body.data.page.pageTotal
            if (_pageStart <= txTotal) {
                queryTxDetailParams.params.crossTxNo =
body.data.list[0].crossTxNo
                //根据跨链编号获取跨链具体详情
                gatewayDeal(detailURL, queryTxDetailParams);
                pageStart += 1;
            }
        }
    })
}

```

```

    }
  }
})
}

function gatewayDeal(_detailURL, _queryTxDetailParams) {
  request({
    url: _detailURL,
    method: "POST",
    json: true,
    headers: "",
    body: _queryTxDetailParams
  }, function (error, response, body) {
    if (!error && response.statusCode == 200) {

      var _txState = body.data.result
      var _txType = body.data.flowList[0].type
      //根据跨链交易详情, 处理跨链交易
      if (_txState == 0 && _txType == 0) {
        //该交易为主链用户发起的跨链交易startTx, 子链网关节点需要发送sendTx
        var params = {"SrcBid":body.data.srcAddress,
          "DestAddress":body.data.destAddress, "SrcChainCode":body.data.srcChainCode,
          "DestChainCode":body.data.destChainCode, "TxType":body.data.payloadType,
          "CrossTxNo":body.data.crossTxNo, "Payload":"","",
          "Extension":body.data.flowList[0].extension, "Remark":body.data.remark,
          "Version": "V1", "Proof":""}
        if (body.data.payloadType == 0) {

          //主链积分转移
          var amount = body.data.payloadGas.amount
          var payload0 = web3.eth.abi.encodeParameter('uint256',
amount)

          params.Payload = payload0

          console.log("user startTx maingas")
        }
        else if (body.data.payloadType == 1)
        {

          var masterAmount = body.data.payloadSgas.masterAmount
          var srcAmount = body.data.payloadSgas.srcAmount
          var srcRate = body.data.payloadSgas.srcTokenRate
          var destAmount = body.data.payloadSgas.destAmount
          var destRate = body.data.payloadSgas.destTokenRate
          var payload1 = web3.eth.abi.encodeParameters(['uint256',
'uint256', 'uint256', 'uint256'], [masterAmount, srcAmount, srcRate,
destAmount, destRate])
          params.Payload = payload1

          console.log("user startTx subSgas")
        }
        else if (body.data.payloadType == 2)
        {

          //var method = body.data.payloadContractCall.contractMethod
          //var para = body.data.payloadContractCall.contractInput

          //var payload2 = web3.eth.abi.encodeParameters(['string',
'bytes', 'bytes'], [method, inputParams, tokenParams])

```

```

    }

    //获取区块号和交易hash
    var ledgerSeq = body.data.flowList[0].lastUpdateSeqNum
    var txHash = body.data.flowList[0].txHash
    var proof = web3.eth.abi.encodeParameters(['uint256', 'string'],
[ledgerSeq, txHash])
    params.Proof = proof

    sendTx(params);
}
else if (_txState == 0 && _txType == 1)
{
    //子链网关节点发送sendAcked
    var params = {"CrossTxNo":body.data.crossTxNo, "Result": 1,
"Version": "v1", "Proof":""}
    var ledgerSeq = body.data.flowList[0].lastUpdateSeqNum
    var txHash = body.data.flowList[0].txHash
    var proof = web3.eth.abi.encodeParameters(['uint256', 'string'],
[ledgerSeq, txHash])
    params.Proof = proof

    console.log("Gateway sendAcked to source chain")
    sendAcked(params)
}
else if (_txState != 0 && body.data.srcChainCode == 0)
{
    //子链网关节点发送sendAcked
    var params = {"CrossTxNo":body.data.crossTxNo, "Result": 1,
"Version": "v1", "Proof":""}
    var ledgerSeq = body.data.flowList[1].lastUpdateSeqNum
    var txHash = body.data.flowList[1].txHash
    var proof = web3.eth.abi.encodeParameters(['uint256', 'string'],
[ledgerSeq, txHash])
    params.Proof = proof

    console.log("Gateway sendAcked to dest chain")
    sendAcked(params)
}
else
{
    console.log("Tx has been solved")
}
}
})
}

```

2.4 系统运行测试说明

本节主要演示星火主链到以太坊链的主链积分转移流程。

2.4.1 启动骨干网网关节点服务

```
cd ~/demo
>node backbone_server_bif.js
//另起终端
cd ~/demo
>node backbone_server_eth.js
```

2.4.2 查询以太坊用户主链积分余额

```
var addr = "0xB2F001cB0a3561C497AcF9aD8879a3A9188a3c4a"
myContract.methods.balanceOf(addr).call({from:addr}, function(error, result){
  console.log(result)
})
~100000
```

2.4.3 发送主链积分转移请求

主链用户发送跨链交易，其账户余额必须够支付要转移的主链积分以及手续费。

获取交易blob:

```
http请求方式: POST
http://{url}/crosschain/user/start/tx/maingas/blob
{
  "accessToken":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhcG1LZXkiOiJkaWQ6Ym1komVmUW9DeEVlcwNObk
VTa05uMlZkcENqWwZwWDE5cWVjIiwiaXNzIjoiaQklGLUNlQUlOIiwiaXhwIjojNjMxNzAwNDkxLCJiaw
QioiJkaWQ6Ym1komVmajdqcjU1VDRsd2ZwRUo5TDFAUXh2d0w0TmdZaFV5In0.uGlglSLKIZygXWcdCaM
8q9PbhbEtF-2Z90xSVD6X0ruI",
  "params": {
    "userId": "did:bid:efRyzZabnVCEbCm4ABNuYwyfSp1bEjwy",
    "srcAddress": "did:bid:efRyzZabnVCEbCm4ABNuYwyfSp1bEjwy",
    "destAddress": "0xB2F001cB0a3561C497AcF9aD8879a3A9188a3c4a",
    "destChainCode": "a202",
    "payload": {
      "amount": "100000000"
    },
    "remark": "",
    "extension": "extension",
    "version": "1000"
  }
}
```

提交交易上链:

```
http请求方式: POST
http://{url}/crosschain/user/start/tx/maingas/submit
{
  "accessToken":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhcG1LZXkiOiJkaWQ6Ym1komVmUW9DeEVlcwNObk
VTa05uMlZkcENqWwZwWDE5cWVjIiwiaXNzIjoiaQklGLUNlQUlOIiwiaXhwIjojNjMxNzAwNDkxLCJiaw
QioiJkaWQ6Ym1komVmajdqcjU1VDRsd2ZwRUo5TDFAUXh2d0w0TmdZaFV5In0.uGlglSLKIZygXWcdCaM
8q9PbhbEtF-2Z90xSVD6X0ruI",
  "params": {
    "blobId": "60",
    "signerList": [
      {

```

```

        "signBlob":
        "bb93db1c223a8d6782120e054e240ace5b3d1521432a9555184af0faa973eca329738a27ea77fc3
        6b2ec88b13eb2bce0248e2807e2727b2505869585e8b35504",
        "publicKey":
        "b06566541eec6e186bca85610308baeff71b7c0372f47a365055da8b585b4b5b4124d1"
    }
}
}
}

```

2.4.4 再次查询以太坊用户主链积分余额

```

var addr = "0xB2F001cB0a3561C497Acf9aD8879a3A9188a3c4a"
myContract.methods.balanceOf(addr).call({from:addr}, function(error, result){
    console.log(result)
})
~100100000

```

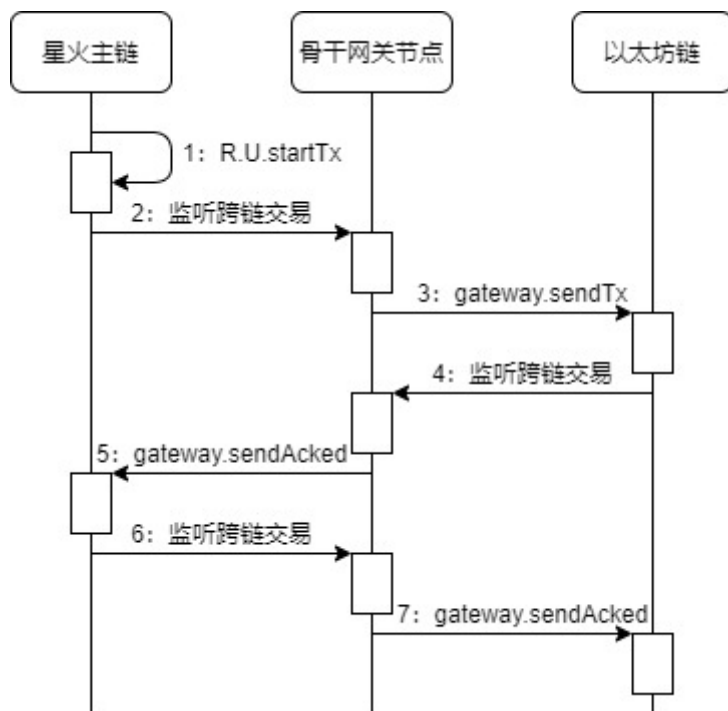
2.4.5 查询主链跨链交易详情

```

{
  "errorCode": 0,
  "message": "操作成功",
  "data": {
    "chainCode": "0",
    "crossTxNo": "0:a202:9742a46b697b062ff01f358aac7ff741",
    "srcChainCode": "0",
    "destChainCode": "a202",
    "srcAddress": "did:bid:efRyzZabnVCEbCm4ABNuYwyfSp1bEjwy",
    "destAddress": "0xB2F001cB0a3561C497Acf9aD8879a3A9188a3c4a",
    "remark": "I am from a202 for start main gas",
    "result": "1",
    "refunded": "0",
    "lastUpdateSeqNum": "72680",
    "payloadType": "0",
    "payloadGas": {
      "amount": "100000000"
    },
    "payloadContractCall": null,
    "payloadSgas": null,
    "payloadData": null,
    "flowList": [
      {
        "chainCode": "0",
        "extension": "extension",
        "version": "1000",
        "type": "0",
        "txSender": "did:bid:efRyzZabnVCEbCm4ABNuYwyfSp1bEjwy",
        "txHash":
        "abf729cc67bf3e5bc340729122215761827f136294ddc2183d92067b14814932",
        "lastUpdateSeqNum": "72678",
        "txErrorCode": "0",
        "txTime": "1632297210000",
        "ack": null
      },
      {

```


2.5 跨链整体流程总结



- 1.星火主链的user转200000主链积分给主链得跨链合约，锁定合约资产；
- 2.网关节点通过超级节点开放平台API监听到跨链交易，并验证跨链交易；
- 3.网关节点转发跨链交易到以太坊链，同时增发主链积分给以太坊的user；
- 4.网关节点通过以太坊的sdk，监听到此跨链交易，并验证跨链交易；
- 5.网关节点给主链发送反馈信息，同时跨链合约将锁定得200000主链积分转给网关节点；
- 6.网关节点监听到此跨链交易，并验证跨链交易；
- 7.网关节点给以太坊发送反馈信息。

三、以太坊跨链合约接口说明

1.设置子链AC码

- 接口：

```
function setChainCode(string memory chainCode) public onlyOwner {}
```

- 权限：合约的owner
- 功能：存储子链注册的AC码，该操作只有部署合约的owner能够调用。
- 参数：

参数	参数类型	描述	是否必须(Y/N)
chainCode	string	子链的AC码	Y

- 结果：若成功，则合约触发event，setChainCodeEvent("setChainCode", chainCode)。

2.设置网关节点地址

- 接口：

```
function setGateway(address gatewayAddress) public onlyOwner {}
```

- 权限：合约的owner
- 功能：注册子链的网关节点地址，该操作只有部署合约的owner能够调用。
- 参数：

参数	参数类型	描述	是否必须(Y/N)
gatewayAddress	address	网关节点的地址	Y

- 结果：若成功，则合约触发event，setGatewayEvent("setGateway", gatewayAddress)。

3.发起跨链交易

- 接口：

```
function startTx(address srcAddress, string memory destBid, string memory srcChainCode, string memory destChainCode, uint8 txType, bytes memory payload, string memory extension, string memory remark, string memory version) public payable {}
```

- 权限：子链内任意需要进行跨链操作的用户
- 功能：用户发起从子链到主链（或者其他子链）的交易，可以是主链积分转移，数据传递，合约互操作，子链积分兑换。
- 参数：

参数	参数类型	描述	是否必须(Y/N)
srcAddress	address	源地址	Y
destBid	string	目标地址	Y
srcChainCode	string	源链的链码	Y
destChainCode	string	目标链的链码	Y
txType	uint	交易类型，0为主链积分转移，1为子链积分兑换，2为数据传递，3为合约互操作；	Y
payload	bytes	json字符串，根据交易类型定义对应的扩展信息，参照跨链类型定义；	Y
extension	string	json字符串，用户扩展信息	N
remark	string	备注信息	N
version	string	版本信息，由原始用户发起交易携带，填写源链跨链合约的版本信息	Y

- 结果：若成功，则合约触发event，startTxEvent("startTx", crossTxNo)。

4.提交跨链交易

- 接口：

```
function sendTx(string memory srcBid, address payable destAddress, string memory srcChainCode, string memory destChainCode, uint8 txType, string memory crossTxNo, bytes memory payload, string memory extension, string memory remark, string memory version, bytes memory proof) public payable {}
```

- 权限：只能由跨链网关进行调用
- 功能：跨链网关提交从主链或者其他子链到当前链的跨链交易。
- 参数：

参数	参数类型	描述	是否必须(Y/N)
srcBid	string	源地址	Y
destAddress	address	目标地址	Y
srcChainCode	string	源链的链码	Y
destChainCode	string	目标链的链码	Y
txType	uint	交易类型，0为主链积分转移，1为子链积分兑换，2为数据传递，3为合约互操作；	Y
crossTxNo	string	跨链交易编号	Y
payload	bytes	json字符串，根据交易类型定义对应的扩展信息，参照跨链类型定义；	Y
extension	string	json字符串，用户扩展信息	N
remark	string	备注信息	N
version	string	版本信息，由原始用户发起交易携带，填写源链跨链合约的版本信息	Y

- 结果：若成功，则合约触发event，sendTxEvent("sendTx", crossTxNo)。

5.提交跨链确认交易

- 接口：

```
function sendAcked(string memory crossTxNo, TxResultEnum result, string memory version, bytes memory proof) public {}
```

- 权限：只能由跨链网关进行调用
- 功能：跨链网关提交从主链或者其他子链到当前链的跨链交易确认，合约在该状态，实现解锁、转移、增发、销毁资产等。
- 参数：

参数	参数类型	描述	是否必须(Y/N)
crossTxNo	string	跨链交易编号	Y
result	uint	跨链交易结果, 0为成功, 1为失败, 2为超时	Y
extension	string	json字符串, 用户扩展信息	N
version	string	版本信息, 由原始用户发起交易携带, 填写源链跨链合约的版本信息	Y

- 结果: 若成功, 则合约触发event, sendAckedEvent("sendAckedTx", crossTxNo)。

6.提取异常的跨链资产

- 接口:

```
function takeOut(string memory crossTxNo, address payable to) public {}
```

- 权限: 发起跨链交易的用户
- 功能: 用户在发起跨链交易异常 (失败或者超时) 后, 可以主动取出资产。
- 参数:

参数	参数类型	描述	是否必须(Y/N)
crossTxNo	string	跨链交易编号	Y
toAddress	uint8	跨链交易结果, 0为成功, 1为失败, 2为超时	Y

- 结果: 若成功, 则合约触发event, takeOutEvent("takeOut", crossTxNo)。

7.授权转移主链积分

- 接口:

```
function approve(address from, address to, uint256 value) public {}
```

- 权限: 子链用户
- 功能: 授权账户 spender 可以从交易发送者账户转出数量为 value 的主链积分。
- 参数:

参数	参数类型	描述	是否必须(Y/N)
from	address	授权账户	Y
to	address	被授权账户	Y
value	uint256	授权能够转移的主链积分数量	Y

- 结果: 若成功, 则合约触发event, approveEvent("approve", value)。

8.转移主链积分

- 接口：

```
function transfer(address to, uint256 value) public {}
```

- 权限：子链用户
- 功能：转移 value 数量的主链积分到的地址 to。
- 参数：

参数	参数类型	描述	是否必须(Y/N)
to	address	接收地址	Y
value	uint256	转移的主链积分数量	Y

- 结果：若成功，则合约触发event，transferEvent("transfer", value)。

9.查询跨链交易

- 接口：

```
function getCrossTx(string memory crossTxNo) public view returns (bytes memory,bytes memory, TxResultEnum, TxRefundedEnum, TxOriginEnum, bytes memory, bytes memory,bytes memory) {}
```

- 权限：子链用户
- 功能：根据跨链交易的编号，查询跨链交易详情。
- 参数：

参数	参数类型	描述	是否必须(Y/N)
crossTxNo	string	跨链交易编号	Y

- 结果：若成功，返回跨链交易详情。

参数	参数类型	描述
basicInfo	bytes	跨链交易的SrcChainCode、DestChainCode、SrcBid、DestBid、TxType等信息
Payload	bytes	跨链交易的扩展信息
Result	uint256	跨链交易结果，0为成功，1为失败，2为超时
Refunded	uint256	跨链交易的资产撤回信息
Origin	uint256	跨链交易的所处位置状态信息
sendProof	bytes	用户发起跨链交易的区块号、交易哈希、验证人信息
ackProof	bytes	网关节点转发跨链交易的区块号、交易哈希、验证人信息
extensionInfo	bytes	remark、extension、version信息

10.查询主链积分余额

- 接口:

```
function balanceOf(address addr) public view returns (uint256) {}
```

- 权限: 子链用户
- 功能: 查询账户的主链积分数量。
- 参数:

参数	参数类型	描述	是否必须(Y/N)
addr	address	带查询账户	Y

- 结果: 若成功, 返回账户主链积分数量。

11.查询跨链合约版本信息

- 接口:

```
function getVersion() public view returns (string memory) {}
```

- 权限: 子链用户
- 功能: 查询跨链合约的版本信息。
- 参数: 无
- 结果: 若成功, 返回合约版本信息。