Hello everyone ,I need some help for transfer ethers using flashbots sendRawTransction on testnet from reactjs localhost.

I am getting the cors error in sendRawTransaction request , can anyone help me out.

Access to fetch at relay-goerli.flashbots from origin localhost:3000 has been blocked by CORS policy: Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

used relay-goerli.flashbots

sendTransaction = (

toAddress,

amount,

chainId,

message = "",

callback = null

) => {

return new Promise(async (resolve, reject) => {

try {

const authSigner = Wallet.createRandom();

let result = null;

// Flashbots provider requires passing in a standard provider

const flashbotsProvider = await FlashbotsBundleProvider.create(

this.PROVIDER, // a normal ethers.js provider, to perform gas estimiations and nonce lookups

authSigner, // ethers.js signer wallet, only for signing request payloads, not transactions

FLASHBOTS_ENDPOINT

// {

// url: FLASHBOTS_ENDPOINT,

// headers:{

// // "Accept":"application/json, application/xml",

// "Content-Type":"application/json",

// "Access-Control-Allow-Origin" : "",

// // "Access-Control-Allow-Credentials": true,

// // "Access-Control-Allow-Methods": "

",

// },

// // fetchOptions:{

// // // mode : "no-cors",

// // // referrer : window.location.origin

// // // "Cache-Control": "no-cache"

// // },

// },

```
);

const block = await this.PROVIDER.getBlock();

const blockBaseFee = Number(block.baseFeePerGas);
const gasLimit = block.gasLimit;
console.log("block gasLimit", gasLimit);
const blockNumber = block.number;
const maxBaseFeeInFutureBlock =
  FlashbotsBundleProvider.getMaxBaseFeeInFutureBlock(blockBaseFee, 6);
const blocksTry = [];
// this.PROVIDER.on("block", async (block) => {
// console.log(`block: ${block}`);
// const blockNumber = block;
let targetBlock = Number(blockNumber);
console.log("targetBlock", targetBlock);
console.log("Starting to run the simulation...");
const amountEther = Web3.utils.toWei(amount.toString(), "ether");
console.log("amont ether", amountEther);
const transactionBundle = [
  {
    signer: this.SIGNER,
    transaction: {
      chainId: chainId,
      type: 2,
      // maxFeePerGas: 300000000000,
      maxFeePerGas: PRIORITY_FEE.add(maxBaseFeeInFutureBlock),
      maxPriorityFeePerGas: PRIORITY_FEE,
      // gasLimit:  gasLimit,
      gasLimit: 3000000,
      value: amountEther,
      data: ethers.utils.hexlify(ethers.utils.toUtf8Bytes(message)),
      to: toAddress,
    },
  },
];
console.log("before sign data", transactionBundle);
const signedTx = await flashbotsProvider.signBundle(
  transactionBundle
);

console.log("signedTx", signedTx);
const sim = await flashbotsProvider.simulate(signedTx, targetBlock + 1);
console.log("sim", sim);
if ("error" in sim || sim.firstRevert !== undefined) {
  console.log("sim error", sim.error.message);
  reject("Transaction failed please try again!");
  // continue;
}
for (let i = 1; i <= 30; i++) {
  blocksTry.push(i);
  console.log("try block count", blocksTry.length);

  if (blocksTry.length >= 30) {
    reject("transaction failed please try again!");
    break;
  }

  // const block = await this.PROVIDER.getBlock(blockNumber)

  // console.log("block",block,"blockNumber",blockNumber);



  const bundleSubmission = await flashbotsProvider.sendRawBundle(
    signedTx,
    targetBlock + i
  );
  console.log("bundleSubmissionbundle", bundleSubmission);
  const bundleResolution = await bundleSubmission.wait();
  console.log("bundleResolutionbundleResolution", bundleResolution);
  if (bundleResolution === FlashbotsBundleResolution.BundleIncluded) {
    console.log(`Congrats, included in ${targetBlock}`);
    result = sim;
    console.log("success transaction", sim);
    resolve(sim);
    break;
    // console.log(JSON.stringify(sim, null, 2));
  } else if (
    bundleResolution ===
    FlashbotsBundleResolution.BlockPassedWithoutInclusion
```

```
  ) {
    console.log(`Not included in ${targetBlock}`);
  } else if (
    bundleResolution === FlashbotsBundleResolution.AccountNonceTooHigh
  ) {
    console.log("Nonce too high, bailing");
    reject("Transaction failed please try again!");
    break;
  }
}

// });
```

} catch (error) { callback && callback(error?.message); reject(error?.message); } });

};

please help me out.