# 242.https://stackoverflow.com/questions/72053763/it-takes-too-long-to-send-transaction-to-the-smart-contract-which-is-already-dep

**T:**It takes too long to send transaction to the smart contract which is already deployed. (Nethereum + Unity)

**Q:**I'm making some dApp using Unity & Nethereum.  
  
I deployed one contract to the Ropsten Test Net using Remix. And I had abi & bytecode of that, so I made Definition & Service C# code using solodity package of VS Code.  
  
I wanted to mint new NFT, and below is the code that I tried.  
  
string url = "my infura - ropsten url";string privateKey = "private Key of my MetaMask account";string userAddress = "public address of my MetaMask account";string contractAddress = "address of deployed contract";var account = new Account(privateKey);var web3 = new Web3(account, url);var service = new MyNFTService(web3, contractAddress);var mintReceipt = await service.MintRequestAndWaitForReceiptAsync(userAddress, "address of metadata");  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
But I can't get receipt even after a long time... Why is this happening? I can't get any answer about that, and I just have to wait.  
  
I have tried everything that I can do, like SendTransactionAndWaitForReceiptAsnyc(), SignAndSendTransaction(), and so on.  
  
The version of Nethereum is 4.1.1, and the version of Unity is 2019.4.21f1.  
  
Below is the part of definition code. (mint)  
  
public partial class MintFunction : MintFunctionBase { }[Function("mint", "uint256")]public class MintFunctionBase : FunctionMessage{ [Parameter("address", "user", 1)] public virtual string User { get; set; } [Parameter("string", "tokenURI", 2)] public virtual string TokenURI { get; set; }}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
And below is the part of service code. (mint)  
  
public Task<string> MintRequestAsync(MintFunction mintFunction){ return ContractHandler.SendRequestAsync(mintFunction);}public Task<TransactionReceipt> MintRequestAndWaitForReceiptAsync(MintFunction mintFunction, CancellationTokenSource cancellationToken = null){ return ContractHandler.SendRequestAndWaitForReceiptAsync(mintFunction, cancellationToken);}public Task<string> MintRequestAsync(string user, string tokenURI){ var mintFunction = new MintFunction(); mintFunction.User = user; mintFunction.TokenURI = tokenURI; return ContractHandler.SendRequestAsync(mintFunction);}public Task<TransactionReceipt> MintRequestAndWaitForReceiptAsync(string user, string tokenURI, CancellationTokenSource cancellationToken = null){ var mintFunction = new MintFunction(); mintFunction.User = user; mintFunction.TokenURI = tokenURI; return ContractHandler.SendRequestAndWaitForReceiptAsync(mintFunction, cancellationToken);}  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]   
  
I am struggle with this problem for five days... Please help me..

1 **Answer**

**A1:**I solved it today! (But I didn't use my service code)  
  
In my opinion, the reason why the transaction didn't work is that the miner can't mine my transaction. (Exactly, they can mine, but they didn't because mining other transaction will give them more money.)  
  
In the document of Netherum, they speak nethereum can set the gas price as the average, but I though it didn't work. After I added a code to estimate and set the gas price, SendRequestAndWaitForReceiptAsync() worked very well. (And I could receive transaction hash.)  
  
Below is the code that I used to solve this problem.  
  
var mintHandler = web3.Eth.GetContractTransactionHandler<MintFunction>();var mint = new MintFunction(){ User = userAddress, TokenURI = "Token URI"};mint.GasPrice = Web3.Convert.ToWei(25, UnitConversion.EthUnit.Gwei);var estimate = await mintHandler.EstimateGasAsync(contractAddress, mint);mint.Gas = estimate.Value;var mintReceipt = await mintHandler.SendRequestAndWaitForReceiptAsync(contractAddress, mint);Debug.Log(mintReceipt.TransactionHash);  
  
WARN: THIS PARAGRAPH CONTAINS TAG: [CODE]