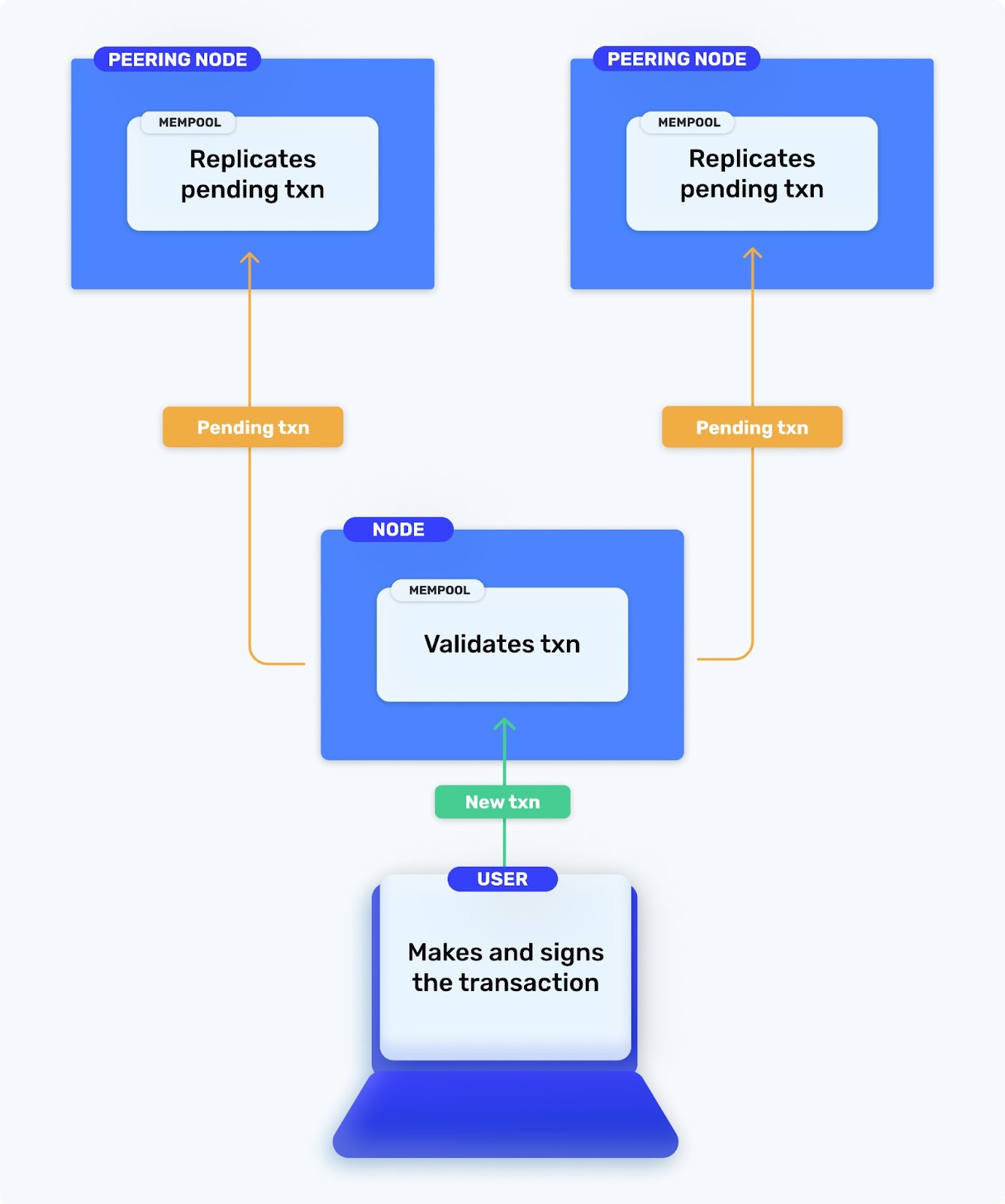
Transactions sent to a node will first land in the mempool, and will remain there until they’re fully confirmed by the network. The mempool handles a few key steps of the process, including:

1. Running a series of validation tests to ensure that the transaction will be successful once submitted to a miner. Some of the validations the node might perform include: Are there available funds to cover the cost of gas? Is the transaction signature valid? Is the nonce next in order for this given sender address?
2. Ordering the list of transactions to be processed based on their gas prices. Higher gas prices mean the transaction will likely be prioritized by the miner.
3. Sharing the pending transactions with the mempools of peered nodes which then continue to forward to other mempools, allowing the entire network to view the list of pending transactions. When a block is mined, the transactions across the entire network with the highest gas prices will be confirmed.

What Does the Ethereum Transaction Flow Look Like?

Now that we’ve covered the mempool, what steps are involved in sending a transaction from a user to permanently storing this transaction on the blockchain?

1. A user generates a new transaction, signs the transaction with their private key, and sends it to a node on the blockchain by calling the method **eth\_sendTransaction** or **eth\_sendRawTransaction** as exposed, typically via a service like [Alchemy’s Supernode API.](https://alchemy.com/supernode?a=a437fb8d33)
2. The transaction is sent to the mempool of a node.
3. The mempool runs a series of validation tests. If the transaction passes these validations it’ll enter a pending state.
4. The node will broadcast the transaction to the other nodes in the blockchain, where it will end up in their corresponding mempools.
5. Once a new block is mined by a node on the blockchain, the ~200 pending transactions with the highest gas prices will be included into the next block on the blockchain. These transactions will be removed from the mempool, and the new block and mempool state propagated through the network
6. After 5 - 7 more blocks are confirmed, the chances of the blockchain re-organizing to follow another fork of blocks (with a different set of confirmed transactions) become vanishingly small. At this point, it’s safe to assume that the transaction is permanently on the blockchain.



What Could Go Wrong With A Pending Ethereum Transaction?

1. The Mempool Might Be Full

Let’s start simple. There’s limits to the size of the mempool (unless you’re using a tool like Alchemy!), both across the entire node and per address in the node. Typically for Geth, it’s 4096 pending transactions, and Geth will drop older pending transactions in favor of newer ones.

2. Your Parameters Might Be Invalid

When you make a transaction, you’ll need to include a variety of relevant parameters, such as:

* What’s the sending address and destination address?
* What’s the value to be transferred?
* How much gas am I willing to use for the transaction?
* What should the current nonce for my transaction be?

If you’ve made a mistake entering these, the node will typically return an error code explaining the issue. If you can’t understand the error message or find the answer via Google, feel free to shoot us a message [on Discord](https://discord.gg/BxRDCHf6) and we’ll try to help you out!

3. Your Gas Price Might Be Too Low

To determine which transactions to prioritize, Ethereum transactions must be submitted with a gas price. Just like a regular auction, a higher gas price means your transaction is more likely to be included into the next block - and too low of a gas price means you might never make it onto the blockchain.

Because the gas price fluctuates rapidly and unpredictably over the course of a day based on demand, it’s very important that you set this field properly. You’ve got a few options here:

* If you’re sending a single transaction manually, you can visit the [**ETH Gas Station website**](https://ethgasstation.info/) to view the most recent gas prices and set your price accordingly.
* If you’re sending requests to a node provider like Alchemy,  the Ethereum JSON-RPC spec exposes an endpoint called [**eth\_estimateGas**](https://docs.alchemy.com/alchemy/apis/ethereum/eth_estimategas) that gives you an estimate of how much gas you’ll need to get your transaction mined! You can use this to automate setting gas prices programmatically.
* If you’d like to receive push notifications when gas prices rise above or fall below a certain price, we provide a tool called [**Gas Price Notifications in Alchemy Notify**](https://docs.alchemy.com/alchemy/guides/using-notify#4-gas-price)that can alert you specifically in those situations! Because it’s a webhook, it’s quite simple to integrate into your code to automatically send transactions when gas prices are low.

4. Your Nonce Might Be Incorrect

Ah, the burning question in everyone’s head - what the heck is a nonce and why do I need it?? Totally valid question and we’re glad you asked.

Quite simply, the nonce is the number of transactions sent from a given address. If you’ve successfully confirmed five transactions (nonces 0-4), the nonce of your next pending transaction will be 5.

Nonces follow a few rules:

* **Nonces must occur in order.** A transaction with nonce 4 will occur strictly before a transaction with nonce 5.
* **Nonces can’t be duplicated or skipped** - there’s exactly one nonce per successful transaction. If you don’t have a transaction with nonce 5, you can’t submit one with nonce 6. SImilarly, you can’t submit two transactions with nonce 5 - only one will go through.

Why do we need them? Nonces are necessary from a cryptographic perspective to ensure that transactions from a given address happen in the order that you sent them, and that your transactions are protected from the [double-spend issue](https://en.wikipedia.org/wiki/Double-spending).

In practice, the nonce is simply a parameter that you must set when you create a transaction. This allows you to have fine-grained manual control over the order of your transactions.

For a real-life example, say you want to buy an NFT from a marketplace and then send a small “finder’s fee” to Bob for helping you discover it. In this case, you’d only want to pay Bob if you successfully bought the NFT. Without nonces, you’d have no way to submit both transactions AND ensure they occurred in a particular order. Using nonces, you can simply label one transaction with nonce N, and one with nonce N + 1!

However, this fine-grained control means you need to submit your transactions with the correct nonce every time! If your transaction is stuck, you’ll need to check your nonce to make sure it’s properly set according to the rules described above.

5. Your Block Might Get Forked

You’ll love this one: even if your transaction successfully is mined, you’re *still* not in the clear 😡 .

Because the blockchain is decentralized, different sets of nodes in the network might have different views of the network. In particular, occasionally two miners will mine a new block at roughly the same time, and peering nodes will hop onto the two new chains at the same rate. In that case, the two new chains will grow in parallel - but eventually one of them will gain just a little more consensus!

When the entire network swaps to one of the chains and abandons the second one, the transactions already “mined” on the old chain become forked and therefore invalid. The forked blocks are also referred to as an “uncle blocks”.

**That means that even after you’ve mined your transaction, it could still be invalidated by a block fork!**You can see a [history of forked blocks on Etherscan here](https://etherscan.io/blocks_forked). Luckily on Ethereum mainnet they’re not very common and typically are under 5 blocks deep, but that’s why your transaction is never “fully” confirmed by a DeFi trading protocol until after 5 - 7 block confirmations minimum.

What Tools Can I Use To Manage My Mempool and Ethereum Pending Transactions?

The Alchemy Mempool Watcher

Say you sent a transaction forty minutes ago to the mempool of a node, but you haven’t seen it confirmed on the blockchain. You’d like some information about its current status, the gas price you sent it with, and the nonce to double-check it. What do you do?

Accessing the state of your mempool transactions was (and still is) a mess. Because the only interface into the status of your mempool is via the Ethereum JSON-RPC API, developers were spending hours querying the various nodes they sent transactions to, merging and sifting through pages of logs, and trying to rebuild the entire state of their mempool by hand - all for a single query.

This is what the average call to a busy mempool looks like:



Have fun reading through this mess.

What’s worse, when developers start sending their requests to multiple nodes for scale, there are serious consistency issues - since transactions originate from  a single node, requests for a particular transaction need to be sticky routed to the same node every time you interface with it.

In response to these challenges, we built the [**Mempool Watcher**](https://dashboard.alchemyapi.io/mempool)- a tool that simplifies a lot of this complexity. First, we handle all the node routing so that even though you might be leveraging dozens of nodes of scale, your mempool always appears as a single entity - with mempool requests properly routed to the correct node within our infrastructure.

Second, we give you some very nice tools to provide a real-time view of the status of your transactions. Transactions are marked as mined, pending, or dropped. Each transaction record includes the duration that the transaction spent pending, the amount of gas required, and access to additional details.