CS4998: Blockchain Development

Primer on Theory

By rjv85@cornell.edu

This document is aimed towards providing a recap of the blockchain theory lecture. It covers the main points of the lecture and provides information that will be useful throughout the course.

Bitcoin

Bitcoin, the first blockchain to become mainstream, was first proposed by Satoshi Nakamoto whose goal was to create "a purely peer-to-peer version of electronic cash [that] would allow online payments to be sent directly from one party to another without going through a financial institution" (Nakamoto, 2008). Although we will not focus on developing Bitcoin scripts in this class, Bitcoin is important to cover because it introduces the concept of a blockchain being an append-only, linked list where each node is a block (hence, the term *blockchain*).

UTXO-Model

Before discussing Ethereum, its good to understand the model that Bitcoin runs on. UTXO, or Unspent Transaction Output, refers to the transaction receipts that serve as the actually tender that is transferred on the Bitcoin network. So as an example, whenever someone states that they have 100 ₿, what this actually means is they have 100 ₿ worth of unspent transaction outputs.

Focusing on how transactions on Bitcoin actually utilize UTXOs, assume Alice has 100 ₿ and decides to send Bob 50 ₿. Since Alice has 100 ₿ worth of UTXOs, she will send 50 ₿ worth of UTXOs to Bob. However, UTXOs cannot be split and this becomes evident if we assume that Alice has 2 UTXOs of 80 ₿ and 20 ₿ each. Alice can utilize the UTXO worth 80 ₿ to pay Bob, but she will have 30 ₿ leftover. Under the UTXO model, this UTXO worth 80 ₿ will turn into the following:

- One UTXO of 50 ₿ , which goes to Bob
- Another UTXO of 30 ₿ , which goes back to Alice

What is Ethereum?

In contrast to Bitcoin, Ethereum is much more encompassing in that it is "the world computer". Ethereum still follows the blockchain model of an append-only linked list of blocks which contain transactions. However, instead of being a blockchain where addresses are tied to their UTXOs and transactions create/delete UTXOs, Ethereum is a computer where each transaction updates the current state of the computer. In addition to accounts holding coins (ether), accounts can also be defined by code. Smart contracts, or accounts with code, are what allows for programs to be developed on Ethereum.

State Model

What is the Ethereum state? Most would point to the current block as being the current state of Ethereum. However, note the following characteristics about Ethereum:

- Blocks do not update the Ethereum blockchain. Transactions are what actually update the blockchain; blocks are simply the containers which hold transactions.

- Ethereum is more than just an accounts-balance mapping. Smart contracts are stateful and because they are stored on the blockchain, are part of the Ethereum state.

A better definition of the Ethereum state would be the characteristics of the storage of the computer at the time of execution. This accounts for the state of Ethereum changing both between blocks (which is obvious) and within blocks (as a result of transcations being executed linearly).

The Ethereum Virtual Machine (EVM) is the computer which is the backbone of the Ethereum state model. We will get into the inner workings of the EVM during the end of the course, but what you need to know is that when we refer to the state of the Ethereum blockchain, we are actually referring to the current characteristics of the EVM's storage at the time of execution.

Storage

Storage refers to the data that is stored on-chain. One instance of storage which all of you have seen is account balances; in Ethereum, balances are stored via a key-pair table which is stored on-chain. Because Ethereum contains smart contracts, the data of these smart contracts also live on-chain. Ignoring the balances of smart contracts, the two most important things that are stored on-chain for smart contracts are the following:

- Contract Runtime Bytecode

- Contract State Variables

A contract's runtime bytecode is the code that gets executed whenever a transaction calls a smart contract. As an example, refer to the following basic smart contract:

```
// SPDX-License-Identifier: MIT
pragma solidity 0.7.0;

contract Basic {}
```

This smart contract does nothing, but if we were to compile it and deploy it to Ethereum, the following bytecode would live forever on the blockchain:

```
0x6080604052600080fdfea26469706673582212-208db385f129f5073a9ca880da8d132b22ebe7b28d
    -738fd57afb2bc1e64660613f64736f6c63430007000033
```

In addition to runtime bytecode, contracts also contain state variables that are also stored on the blockchain. Consider the following smart contract:

```
// SPDX-License-Identifier: MIT
pragma solidity 0.7.0;

contract Basic {

    uint256 num = 1;

}
```

Because the state variable `num` is nonzero and `num` is part of the smart contract's state, it too must be stored on the blockchain.

Gas

The Ethereum Virtual Machine allows for smart contracts to serve as programs which anyone can interact with. However, the EVM was also designed so that malicious actors wouldn't be able to bring down the network via attacks. It is for this reason that concept of gas exists; gas is a unit of computation that is assigned to each EVM operation. Note that operations refer to the machine level instruction that the EVM is executing, and we will get into this during lectures on the EVM. At a surface level though, each transaction utilizes a certain amount of computation work which is denominated in gas, which a user must pay for with ether. The following terms become clear now:

- Gas limit: the maximum amount of gas the EVM is allowed to use in a transaction. If the EVM goes over this limit, the transaction is reverted
- Gas Fee: the amount of ether (denominated in gwei, $1 * 10^{-9}$ ether) one is paying per unit of gas

Because the execution of a transaction is dependent on the current state of the EVM, this means that the gas usage of a transaction, estimated prior to its execution, can differ from the actual amount of gas used in a transaction. It is for this reason that gas limit is specified in a transaction.