Foundations of FinTech

Blockchain

Eshwar Venugopal

UCF

Simple Blockchain

Simple Blockchain

- Recollect that a blockchain is:
  1. A database (ledger of activities)
  2. Distributed across multiple servers (transparent)
  3. Immutable & Consensus based (tamper-proof to a certain extend)

- In the toy versions, we are going to implement (1) & (3)
  - We will be implementing a simpler (and different) proof-of-work algorithm
  - We will NOT implement Merkle Tree data structures or worry about verifications
  - We will NOT implement a multi-node network system

Simple Blockchain: Version 1

- Let us divide the coding into the following core functions:
    1. Define how a block would look like
    2. Define the hashing function
    3. Define a process to add blocks
    4. A process to listen for transactions

- Refer to “demoChain_v1.py” file

Simple Blockchain: Version 1

1. Define how a block would look like
   - Recollect that a block contains:
     - Previous block hash
     - Data
     - Timestamp

```
class Block:
  def __init__(self, index, timestamp, data, previous_hash):
    self.index = index
    self.timestamp = timestamp
    self.data = data
    self.previous_hash = previous_hash
    self.hash = self.hash_block()
```

Simple Blockchain: Version 1

2. Define the hashing function
   - We will use the SHA256 hash function
   - Take the header data and hash it twice

```
def hash_block(self):
    header=(str(self.index) + str(self.timestamp) + str(self.data) + str(self.previous_hash)).encode()
    inner_hash=hasher.sha256(header).hexdigest().encode()
    outer_hash=hasher.sha256(inner_hash).hexdigest()
    return outer_hash
```

Simple Blockchain: Version 1

3. Define a process to add blocks
   a) At the beginning we need to add a genesis block

```
def create_genesis_block():
  # Manually construct a block with
  # index zero and arbitrary previous hash
  return Block(0, date.datetime.now(), "Genesis Block", "0")
```

Simple Blockchain: Version 1

3. Define a process to add blocks
   b) Adding new blocks:
      - We need the current data for the new block
      - We need the hash value of the previous block
      - We return the block

```
def next_block(last_block):
  this_index = last_block.index + 1
  this_timestamp = date.datetime.now()
  this_data = input("Hey! Input your data for block #{}: ".format(this_index)) + str(this_index)
  this_hash = last_block.hash
  return Block(this_index, this_timestamp, this_data, this_hash)
```

Simple Blockchain: Version 1

4. A process to listen for transactions
   - We are going to create the genesis block and listen for transactions using a simple loop

```
# Create the blockchain and add the genesis block
blockchain = [create_genesis_block()]
previous_block = blockchain[0]

# How many blocks should we add to the chain after the genesis block
num_of_blocks_to_add = 5

# Add blocks to the chain
for i in range(0, num_of_blocks_to_add):
  block_to_add = next_block(previous_block)
  blockchain.append(block_to_add)
  previous_block = block_to_add
  # Tell everyone about it!
  print ("Block #{} has been added to the blockchain!".format(block_to_add.index))
  print ("Hash: {}\n".format(block_to_add.hash))
```

Simple Blockchain: Version 2

- In this version:
  - Add a server to listen for transactions
  - Add a node
  - Add consensus algorithm
  - Add mining functionality

- This version does NOT support:
  - Multiple nodes across different computers
  - Transactions & verifications across parties

- Refer to “demoChainServer_v1.py” file

Simple Blockchain: Version 2

Command to add a transaction:

curl "http://localhost:5000/txion" -d "{\"from\": \“Alice\", \"to\":\“Bob\", \"amount\": 300}" -H "Content-Type: application/json"

Command to mine a block:

curl localhost:5000/mine