НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт з виконання кваліфікаційного дослідження

РОЗГОРТАННЯ СИСТЕМ ETHEREUM ТА КРИПТОВАЛЮТ

Виконали студенти
групи ФІ-32мн
Баєвський Констянтин,
Шифрін Денис,
Кріпака Ілля

Київ — 2024

ЗМІСТ

ВСТУП

0.1 Мета практикуму

Отримати навички налаштування платформ виконання смарт–контрактів та криптовалют.

0.1.1 Постановка задачі та варіант

Треба виконати	**Зроблено**
Провести налаштування обраної системи та виконати тестові операції в системі.	✓
Порівняти особливості розгортання різних систем криптовалют із системою Ethereum.	✓

0.2 Хід роботи/Опис труднощів

На початку роботи над практикумом вибрали гуртом 2 варіант. Згідно вибраного варіанту у даній роботі буде продемонстровано спробу запуску, налаштування системи Ethereum та виконання тестових операцій в ній. Також наведено короткий порівняльний аналіз даної системи з іншими.

1 НАЛАШТУВАННЯ СИСТЕМИ ETHEREUM

Для налаштування приватної ноди мережі Ethereum на основі протоколу Proof-Of-Work (PoW) з використанням Geth, будемо використовувати PoW алгоритм — Ethash. Цей протокол є останньою версією алгоритму Dagger-Hashimoto, який використовується для майнінгу.

1.1 Запуск та налаштування приватної ноди

Спочатку встановимо geth:

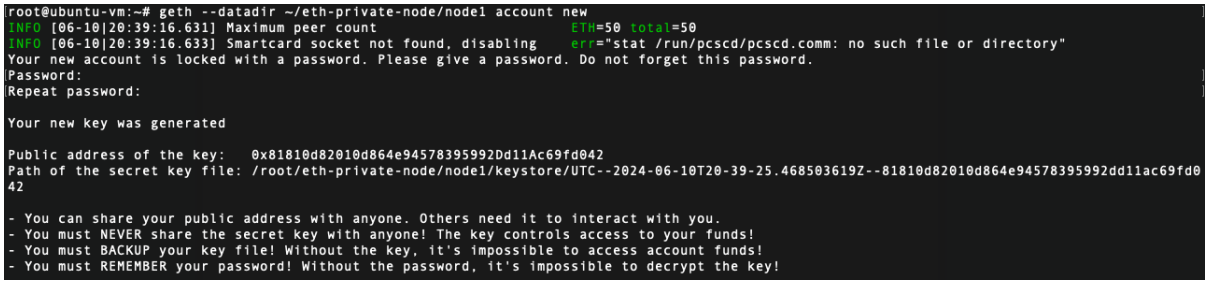
```
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install geth
```

Тепер створимо директорію для нод:

```
mkdir -p ~/eth-private-node/{node1,node2,shared}
```

Далі створимо два облікових записи:

```
geth --datadir ~/eth-private-node/node1 account new
```



Для 1-го акаунту отримали:

Public address of the key: `0x81810d82010d864e94578395992Dd11Ac69fd042`

```
geth --datadir ~/eth-private-node/node2 account new
```

```
root@ubuntu-vm:~# geth --datadir ~/eth-private-node/node2 account new
INFO [06-10|20:40:44.384] Maximum peer count                       ETH=50 total=50
INFO [06-10|20:40:44.387] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0xe097FE6ca37531b2c48B71707eBfc765eB7EaCb2
Path of the secret key file: /root/eth-private-node/node2/keystore/UTC--2024-06-10T20-40-52.449149424Z--e097fe6ca37531b2c48b71707ebfc765eb7eacb2

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

Для 2-го акаунту отримали:

Public address of the key: `0xe097FE6ca37531b2c48B71707eBfc765eB7EaCb2`

Створимо файл `genesis.json` у директорії `~/eth-poa/shared`:

```
nano ~/eth-private-node/shared/genesis.json
```

з наступним вмістом:

```
 GNU nano 7.2                                /root/eth-private-node/shared/genesis.json *
{
  "config": {
    "chainId": 1337,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "alloc": {
    "81810d82010d864e94578395992Dd11Ac69fd042": {
      "balance": "1000000000000000000"
    },
    "e097FE6ca37531b2c48B71707eBfc765eB7EaCb2": {
      "balance": "5000000000000000000"
    }
  },
  "difficulty": "0x400",
  "gasLimit": "0x8000000",
  "alloc": {}
}
```

Тепер переглянемо, що маємо в нашій структурі папки ноди:

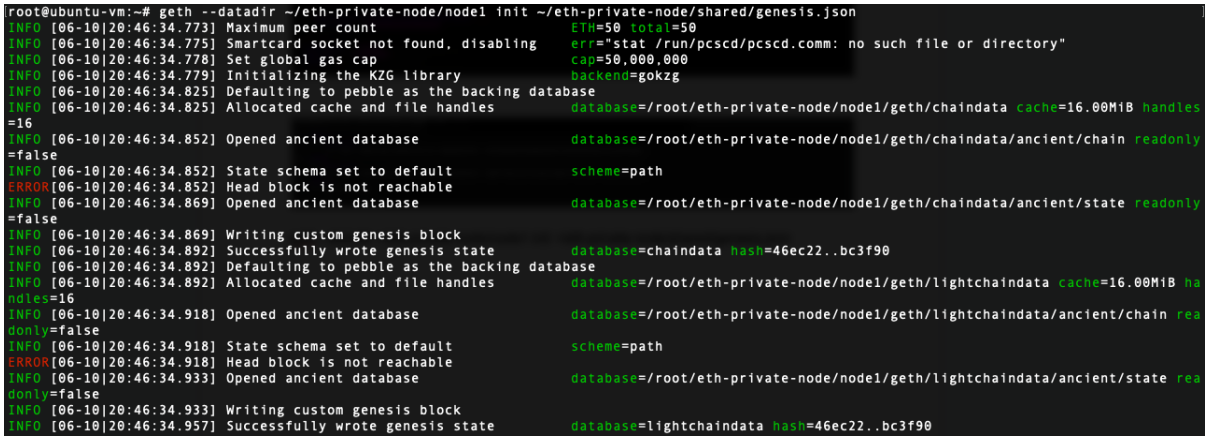
```
tree -L 3
```

```
root@ubuntu-vm:~/eth-private-node# tree -L 3
.
├── node1
│   └── keystore
│       └── UTC--2024-06-10T20-39-25.468503619Z--81810d82010d864e94578395992dd11ac69fd042
├── node2
│   └── keystore
│       └── UTC--2024-06-10T20-40-52.449149424Z--e097fe6ca37531b2c48b71707ebfc765eb7eacb2
└── shared
    └── genesis.json

6 directories, 3 files
```

Далі ініціалізуємо ноди.

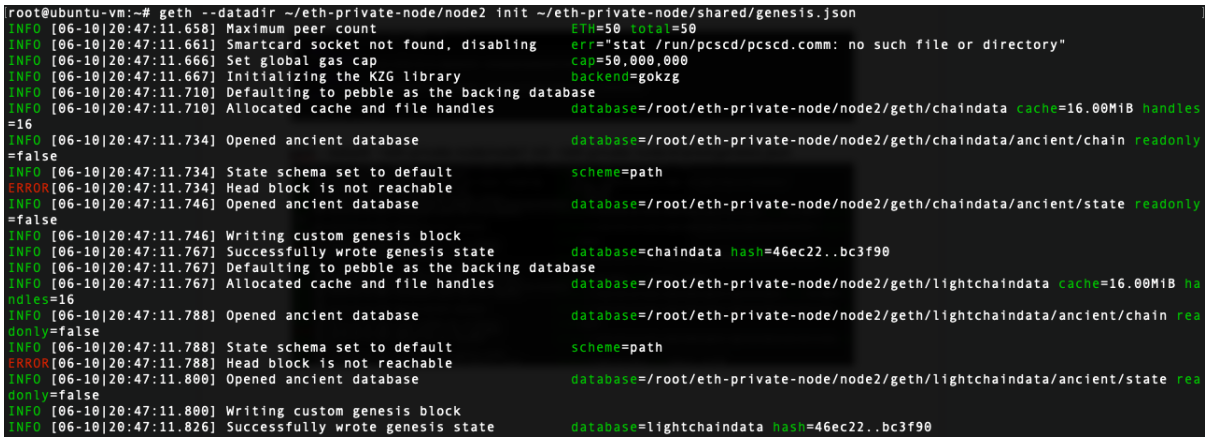
Ініціалізуємо першу ноду з genesis-файлом:

```
geth --datadir ~/eth-private-node/node1 init
~/eth-private-node/shared/genesis.json
```



Ініціалізуємо другу ноду з genesis-файлом:

```
geth --datadir ~/eth-private-node/node2 init
~/eth-private-node/shared/genesis.json
```



Тепер запустимо першу ноду в ролі валідатора:

```
geth --datadir ~/eth-private/node1 --http --http.addr "0.0.0.0"
--http.port 8545 --http.api "eth,net,web3,personal"
--nodiscover console
```

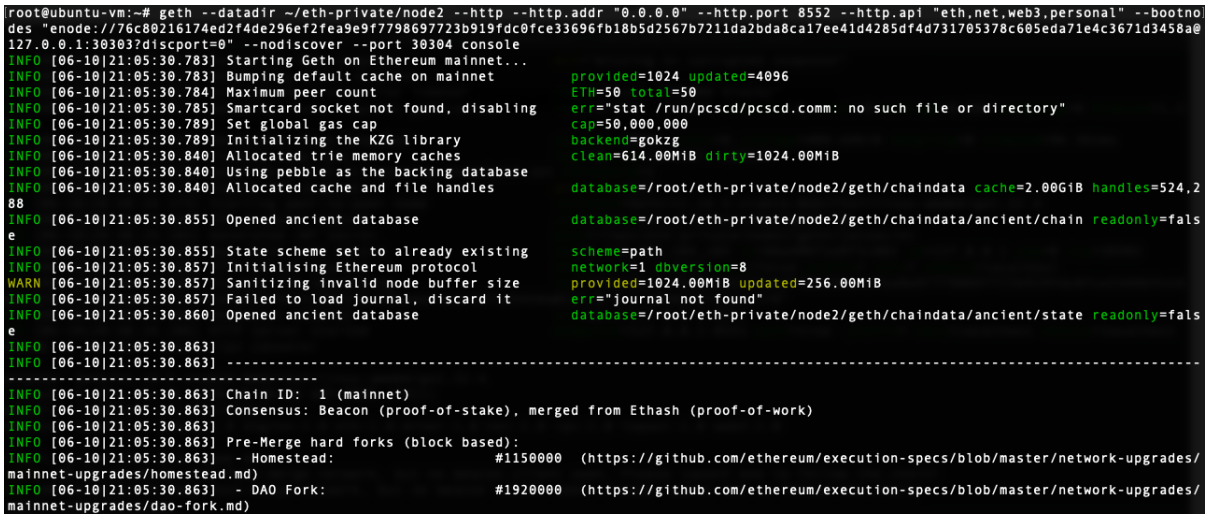
```
root@ubuntu-vm:~# geth --datadir ~/eth-private/node1 --http --http.addr "0.0.0.0" --http.port 8545 --http.api "eth,net,web3,personal" --nodisc
over console
INFO [06-10|20:48:23.419] Starting Geth on Ethereum mainnet...
INFO [06-10|20:48:23.419] Bumping default cache on mainnet         provided=1024 updated=4096
INFO [06-10|20:48:23.423] Maximum peer count                       ETH=50 total=50
INFO [06-10|20:48:23.425] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [06-10|20:48:23.430] Set global gas cap                       cap=50,000,000
INFO [06-10|20:48:23.430] Initializing the KZG library             backend=gokzg
INFO [06-10|20:48:23.480] Allocated trie memory caches             clean=614.00MiB dirty=1024.00MiB
INFO [06-10|20:48:23.481] Defaulting to pebble as the backing database
INFO [06-10|20:48:23.481] Allocated cache and file handles         database=/root/eth-private/node1/geth/chaindata cache=2.00GiB handles=524,2
88
INFO [06-10|20:48:23.513] Opened ancient database                  database=/root/eth-private/node1/geth/chaindata/ancient/chain readonly=fals
e
INFO [06-10|20:48:23.513] State schema set to default              scheme=path
INFO [06-10|20:48:23.513] Initialising Ethereum protocol           network=1 dbversion=<nil>
ERROR[06-10|20:48:23.513] Head block is not reachable
WARN [06-10|20:48:23.516] Sanitizing invalid node buffer size      provided=1024.00MiB updated=256.00MiB
INFO [06-10|20:48:23.533] Opened ancient database                  database=/root/eth-private/node1/geth/chaindata/ancient/state readonly=fals
e
INFO [06-10|20:48:23.533] Writing default main-net genesis block
INFO [06-10|20:48:24.073]
INFO [06-10|20:48:24.073] ----------------------------------------------------------------------------------------------------------------
-----------------------------------
INFO [06-10|20:48:24.073] Chain ID:  1 (mainnet)
INFO [06-10|20:48:24.073] Consensus: Beacon (proof-of-stake), merged from Ethash (proof-of-work)
INFO [06-10|20:48:24.073]
INFO [06-10|20:48:24.073] Pre-Merge hard forks (block based):
INFO [06-10|20:48:24.073]  - Homestead:                    #1150000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/homestead.md)
INFO [06-10|20:48:24.073]  - DAO Fork:                     #1920000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/dao-fork.md)
INFO [06-10|20:48:24.073]  - Tangerine Whistle (EIP 150): #2463000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/tangerine-whistle.md)
INFO [06-10|20:48:24.073]  - Spurious Dragon/1 (EIP 155): #2675000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/spurious-dragon.md)
INFO [06-10|20:48:24.073]  - Spurious Dragon/2 (EIP 158): #2675000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/spurious-dragon.md)
INFO [06-10|20:48:24.073]  - Byzantium:                    #4370000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/byzantium.md)
INFO [06-10|20:48:24.073]  - Constantinople:               #7280000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/constantinople.md)
INFO [06-10|20:48:24.073]  - Petersburg:                   #7280000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/petersburg.md)
INFO [06-10|20:48:24.073]  - Istanbul:                     #9069000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/istanbul.md)
INFO [06-10|20:48:24.073]  - Muir Glacier:                 #9200000  (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/muir-glacier.md)
INFO [06-10|20:48:24.073]  - Berlin:                       #12244000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/berlin.md)
INFO [06-10|20:48:24.073]  - London:                       #12965000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/london.md)
INFO [06-10|20:48:24.073]  - Arrow Glacier:                #13773000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/arrow-glacier.md)
INFO [06-10|20:48:24.073]  - Gray Glacier:                 #15050000 (https://github.com/ethereum/execution-specs/blob/master/network-upgrades/
mainnet-upgrades/gray-glacier.md)
INFO [06-10|20:48:24.073]
INFO [06-10|20:48:24.073] Merge configured:
INFO [06-10|20:48:24.073]  - Hard-fork specification:    https://github.com/ethereum/execution-specs/blob/master/network-upgrades/mainnet-upgr
ades/paris.md
INFO [06-10|20:48:24.073]  - Network known to be merged: true
INFO [06-10|20:48:24.073]  - Total terminal difficulty:  58750000000000000000
INFO [06-10|20:48:24.073]
INFO [06-10|20:48:24.073] Post-Merge hard forks (timestamp based):
INFO [06-10|20:48:24.073]  - Shanghai:                     @1681338455 (https://github.com/ethereum/execution-specs/blob/master/network-upgrade
s/mainnet-upgrades/shanghai.md)
INFO [06-10|20:48:24.073]  - Cancun:                       @1710338135 (https://github.com/ethereum/execution-specs/blob/master/network-upgrade
s/mainnet-upgrades/cancun.md)
INFO [06-10|20:48:24.073]
INFO [06-10|20:48:24.073] ----------------------------------------------------------------------------------------------------------------
-----------------------------------
INFO [06-10|20:48:24.073]
INFO [06-10|20:48:24.073] Loaded most recent local block           number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=55y2mo3w
WARN [06-10|20:48:24.073] Failed to load snapshot                  err="missing or corrupted snapshot"
INFO [06-10|20:48:24.075] Rebuilding state snapshot
INFO [06-10|20:48:24.077] Initialized transaction indexer          range="last 2350000 blocks"
INFO [06-10|20:48:24.077] Resuming state snapshot generation       root=d7f897..0f0544 accounts=0 slots=0 storage=0.00B dangling=0 elapsed=1.2
73ms
INFO [06-10|20:48:24.170] Generated state snapshot                 accounts=8893 slots=0 storage=409.64KiB dangling=0 elapsed=94.361ms
INFO [06-10|20:48:24.170] Enabled snap sync                        head=0 hash=d4e567..cb8fa3
INFO [06-10|20:48:24.171] Gasprice oracle is ignoring threshold set threshold=2
WARN [06-10|20:48:24.173] Engine API enabled                       protocol=eth
INFO [06-10|20:48:24.173] Starting peer-to-peer node               instance=Geth/v1.14.5-stable-0dd173a7/linux-amd64/go1.22.4
INFO [06-10|20:48:24.184] IPC endpoint opened                      url=/root/eth-private/node1/geth.ipc
INFO [06-10|20:48:24.185] Generated JWT secret                     path=/root/eth-private/node1/geth/jwtsecret
INFO [06-10|20:48:24.185] New local node record                    seq=1,718,052,504,184 id=68a40072e0f5c083 ip=127.0.0.1 udp=0 tcp=30303
INFO [06-10|20:48:24.185] HTTP server started                      endpoint=[::]:8545 auth=false prefix= cors= vhosts=localhost
INFO [06-10|20:48:24.185] Started P2P networking                   self="enode://76c80216174ed2f4de296ef2fea9e9f7798697723b919fdc0fce33696fb18
b5d2567b7211da2bda8ca17ee41d4285df4d731705378c605eda71e4c3671d3458a@127.0.0.1:30303?discport=0"
INFO [06-10|20:48:24.186] WebSocket enabled                        url=ws://127.0.0.1:8551
INFO [06-10|20:48:24.186] HTTP server started                      endpoint=127.0.0.1:8551 auth=true  prefix= cors=localhost vhosts=localhost
Welcome to the Geth JavaScript console!

instance: Geth/v1.14.5-stable-0dd173a7/linux-amd64/go1.22.4
at block: 0 (Thu Jan 01 1970 00:00:00 GMT+0000 (UTC))
 datadir: /root/eth-private/node1
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> WARN [06-10|20:48:59.174] Post-merge network, but no beacon client seen. Please launch one to follow the chain!
```

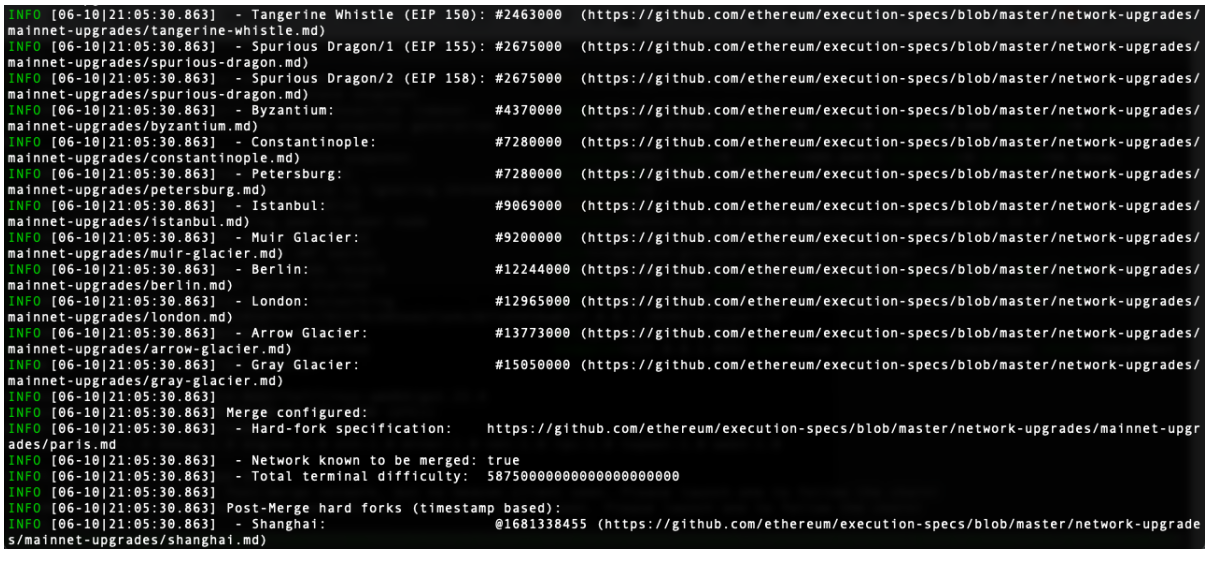
Тепер запустимо другу ноду та підключимо її до першої як до бут-ноди. Для цього спочатку отримаємо enode першої ноди:

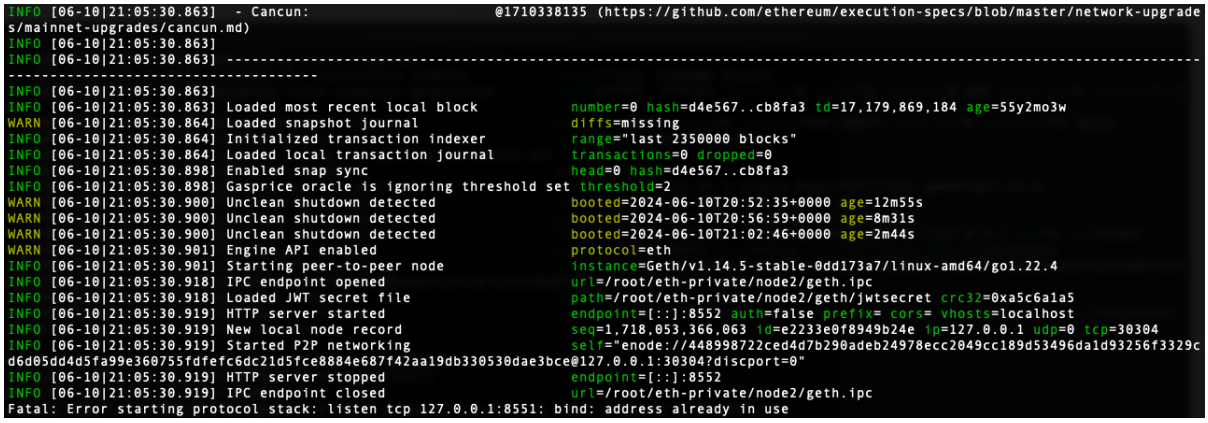
```
admin.nodeInfo.enode
```

Далі запустимо другу ноду, вказавши enode першої ноди:

```
geth --datadir ~/eth-private/node2 --http --http.addr "0.0.0.0"
--http.port 8552 --http.api "eth,net,web3,personal" --bootnodes
"enode://76c8021617.." --nodiscover --port 30304 console
```







Отримуємо помилку.

Спробуємо також просто запустити другу ноду, яка ніяк не залежить від першої:

```
geth --datadir ~/eth-private/node2 --http --http.addr "0.0.0.0"
--http.port 8552 --http.api "eth,net,web3,personal" --nodiscover
--port 30304 console
```

Бачимо, що отримуємо помилку аналогічну першому випадку.

1.2 Тестування операцій в приватній ноді

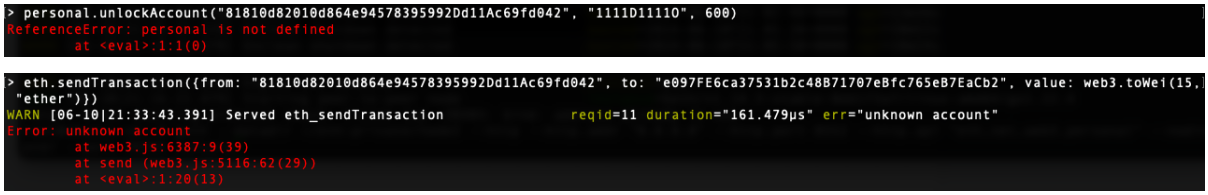
Спробуємо в запущеній першій ноді перевірити її стан:

```
eth.syncing
```


```
eth.blockNumber
```

Спробуємо перевірити стан майнінгу ноди:

А також перевіримо можливість переведення ETH між акаунтами:



В обох випадках отримали відмову перевірки, оскільки не був запущений майнінг для даної ноди, а відповідно – недоступні будь-які операції.

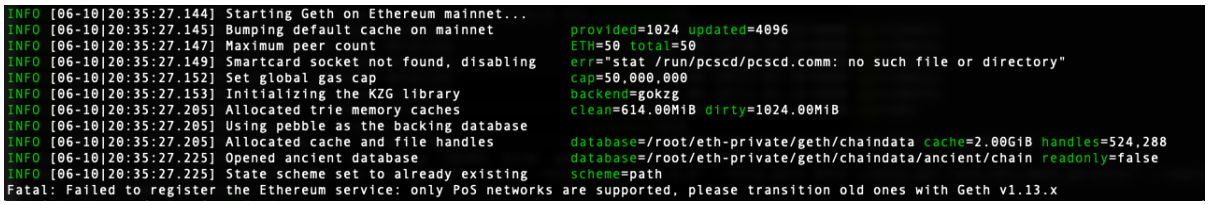
При спробі виконати команду запуску ноди з майнінгом з оператором `-miner.threads`:

```
geth --datadir ~/eth-private-node/node1 --http --http.addr "0.0.0.0"
--http.port 8545 --http.corsdomain "*" --http.api personal,eth,net,
web3,miner --mine --miner.threads 1 --networkid 1337
--allow-insecure-unlock
```

А також при спробі ж виконати команду запуску ноди – з оператором `-miner.etherbase`:

```
geth --datadir ~/eth-private-node/node1 --networkid 1337 --http
--http.addr "0.0.0.0" --http.port 8545 --http.corsdomain "*"
--http.api personal,eth,net,web3,clique --allow-insecure-unlock
--nodiscover --mine --miner.etherbase "81810d8201...Dd11Ac69fd042"
--unlock "81810d8201...Dd11Ac69fd042"
--password /root/eth-private-node/.. --verbosity 3 console
```

В обох випадках отримували помилку:



Крім цього, пробували запуск та налаштування власної ноди Ethereum на протоколі PoA (Proof-Of-Authority), де отримували аналогічну помилку про недоступність Geth до відмінних від PoS протоколів.

Висновки до розділу 1

Отже, наразі більшість функціоналу Geth для PoW є застарілою, оскільки майнінг був вимкнений при переході на Ethereum 2.0 на Proof-Of-Stake. А PoS потребує більш складних зусиль (включно зі смарт-контрактом для стейкінгу) для коректного налаштування та роботи приватної ноди Ethereum.

2 ПОРІВНЯЛЬНИЙ АНАЛІЗ СИСТЕМИ ETHEREUM З ІНШИМИ СИСТЕМАМИ

Системи криптовалют мають різні архітектури, протоколи та механізми, що впливають на процес їх розгортання. Далі проведемо аналіз, де порівняємо особливості розгортання систем Bitcoin, Litecoin, Dash, NEO та Ethereum з метою визначити можливість взаємозаміни модулів різних систем.

2.1 Порівняння основних характеристик

– Розгортання та налаштування

1) **Bitcoin** використовує механізм Proof of Work (PoW) з алгоритмом хешування SHA-256. Розгортання Bitcoin-нод включає завантаження клієнта Bitcoin Core, синхронізацію з мережею та налаштування конфігураційного файлу для мережі P2P.

2) **Litecoin** подібний до Bitcoin, але використовує алгоритм хешування Scrypt, що спрощує майнінг для звичайних користувачів. Розгортання Litecoin-нод аналогічне до Bitcoin і передбачає встановлення клієнта Litecoin Core.

3) **Dash** додає до PoW систему Masternodes, що забезпечує додаткові функції, такі як миттєві транзакції та підвищена конфіденційність. Розгортання Dash-нод включає налаштування стандартних нод та Masternodes з додатковими параметрами конфігурації.

4) **NEO** використовує механізм децентралізованого візантійського Fault Tolerance (dBFT). Розгортання NEO-нод включає налаштування NEO VM та підтримку смарт-контрактів, що вимагає додаткового налаштування мережевих параметрів і конфігурації вузлів.

5) **Ethereum** використовує PoW з алгоритмом Ethash і підтримує

смарт-контракти через Ethereum Virtual Machine (EVM). Розгортання Ethereum-нод передбачає встановлення клієнта Geth або Parity, налаштування RPC та синхронізацію з мережею.

6) **Ethereum 2.0** використовує PoS з можливістю розгортання за допомогою клієнтів Prysm, Teku, Lighthouse або Nimbus, налаштування RPC та синхронізації з мережею. Також використовується новий блокчейн – Beacon Chain, який керує консенсусом мережі PoS, а в майбутньому планується використовувати шардинг для підвищення масштабованості.

– Аналіз взаємозаміни модулів

Через різні архітектури та протоколи, взаємозаміна модулів між різними системами є складною та часто неможливою. Наведемо основні причини:

1) **Алгоритми хешування:** кожна система використовує свій власний алгоритм хешування (SHA-256 для Bitcoin, Scrypt для Litecoin, X11 для Dash, SHA-3 для NEO, Ethash для Ethereum). Це робить неможливою взаємозаміну цих компонентів без значних змін у протоколі.

2) **Механізми консенсусу:** відмінності між PoW (Bitcoin, Litecoin), PoS (Ethereum 2.0), і dBFT (NEO) роблять механізми консенсусу несумісними. Кожен механізм вимагає специфічної логіки для підтвердження транзакцій та генерації блоків.

3) **Віртуальні машини:** Ethereum використовує EVM(Ethereum Virtual Machine) для виконання смарт-контрактів, тоді як NEO використовує NEO VM. Віртуальні машини розроблені для виконання контрактів у специфічних середовищах, що ускладнює їх взаємозаміну.

4) **Модульність коду:** Bitcoin має відносно монолітну архітектуру, що робить його важчим для модифікації. Ethereum та NEO мають більш модульний підхід, що дозволяє легше додавати або змінювати функціональні модулі. Однак, різні архітектурні рішення все одно ускладнюють взаємозаміну модулів між системами.

Характеристика	Bitcoin	Litecoin	Dash	NEO	Ethereum	Eth 2.0
Consensus	PoW	PoW	PoW *	dBFT	PoW	PoS
Hash function	SHA-256	Scrypt	X11	SHA-3	Ethash	n/a
Block time	10 хв	2.5 хв	2.5 хв	15-25 с	12-15 с	12-15 с
Modularity	Низька	Середня	Середня	Висока	Висока	Висока
Virtual machine	Ні	Ні	Ні	NEO VM	EVM	EVM
Smart-contracts	Ні	Ні	Обмежено	Так	Так	Так
DApps	Ні	Ні	Обмежено	Так	Так	Так
Mining	Так	Так	Так	Ні	Так	Ні
Staking	Ні	Ні	Так *	Так	Ні	Так
TPS	7	56	28	1000+	30	100,000+
Fees	Високі	Низькі	Середні	Низькі	Високі	Низькі

Таблиця 2.1 – Порівняння основних характеристик криптовалютних систем

* — DASH надає можливість використання Masternodes: повноцінних вузлів, які стимулюються отриманням частини винагороди за блок в обмін на завдання, які вони виконують для мережі, серед яких найважливішими є участь у транзакціях.

1) **Порівняння алгоритмів консенсусу**

– PoS та dBFT забезпечують кращу енергоефективність та масштабованість.

– PoW залишається безпечним, але енергоємним і менш ефективним в плані масштабованості.

2) **Порівняння алгоритмів гешування**

– SHA-256 і SHA-3 є добре перевіреними алгоритмами з високою безпекою та ефективністю.

– X11 використовує комбінацію 11 різних хеш-функцій, що забезпечує високу безпеку.

– Scrypt є менш енергоефективним, ніж SHA-256, але забезпечує кращий захист від ASIC-майнерів.

– Ethash забезпечує адекватну безпеку, але має деякі недоліки в енергоефективності.

– Eth 2.0 не використовує хешування для консенсусу, але система PoS

забезпечує високу безпеку.

3) **Порівняння за середнім часом блоку**

– Швидший час блоку ($\leqslant$ 15 секунд) дозволяє швидше підтверджувати транзакції, що є критично важливим для багатьох застосунків.

– Повільніший час блоку ($>$ 5 хвилин) забезпечує більше безпеки, але знижує швидкість підтвердження транзакцій.

4) **Модульність системи**

– Висока модульність дозволяє легше адаптувати та інтегрувати модулі.

– Через різні архітектурні рішення та протоколи взаємозаміна модулів між різними системами є складною.

5) **Наявність віртуальних машин**

– Надається підтримку смарт-контрактів, що дозволяє створювати децентралізовані додатки (dApps), які розширюють функціонал мережі (Ethereum, NEO).

– Надається можливість оновлень: віртуальні машини, такі як EVM, дозволяють легше впроваджувати оновлення та зміни.

– Відсутність віртуальної машини обмежує можливості системи до простих транзакцій без складних програмованих логік.

6) **Порівняння за кількістю транзакцій на секунду (TPS)**

– Високі TPS ($>$ 1000 TPS) важливі для масштабованості та підтримки великої кількості користувачів.

– Низькі TPS ($\leqslant$ 50 TPS) обмежують пропускну здатність мережі, що може призводити до затримок та підвищених комісій.

Висновки до розділу 2

Розгортання кожної системи криптовалют має свої унікальні особливості, що обмежує можливість взаємозаміни модулів. Враховуючи різні алгоритми хешування, механізми консенсусу, віртуальні машини та рівні модульності коду, інтеграція модулів між різними системами потребує значних змін та адаптацій.

Таким чином, ефективне розгортання та підтримка криптовалютних систем вимагають урахування їх унікальних особливостей та обмежень, а також чіткої архітектурної інтеграції для досягнення необхідної функціональності. Але, якщо вам потрібна система з алгоритмом консенсусу PoS (без можливості майнінгу) – то Ethereum 2.0 є одним з кращих варіантів.

ВИСНОВКИ

У ході даної лабораторної роботи було отримано навички налаштування платформ виконання смарт-контрактів та криптовалют. Зокрема проведено спробу налаштування, запуску та тестування власної ноди мережі Ethereum. Крім цього, було детально проведено порівняльний аналіз системи Ethereum з іншими системами криптовалют.