Bitcoin Core and Lightning Network Daemon (LND) Setup Documentation

SK Golam Kuddus
Roll Number: CRS2318

December 16, 2024

Contents

1 Introduction **4**

2 Differences Between Mainnet, Testnet, and Regtest **4**

3 Installing Bitcoin Core **5**
3.1 Download and Extract Bitcoin Core 5
3.2 Verify Installation 5

4 Configuring Bitcoin Core **5**
4.1 Edit Configuration File 5
4.2 Start Bitcoin Core 5

5 Installing Go and LND **6**
5.1 Install Go 6
5.2 Install LND 6

6 Configuring Lightning Nodes **7**
6.1 Set Up Directories 7
6.1.1 Verify directories 7
6.2 Create Configuration Files 7
6.2.1 Node A Configuration 7
6.2.2 Node B Configuration 9
6.2.3 Node C Configuration 9
6.2.4 Node D Configuration 10
6.3 Start Each Node 10

7 Wallet Setup and Funding **11**
7.1 Unlock Wallets 13
7.2 Retrieve Public Key and Node Information 13
7.3 Generate a New Wallet Address for Each Node 15

8 Bitcoin Commands and Explanations 16
8.1 Create a New Wallet . . . 16
8.2 Generate a New Bitcoin Address . . . 16
8.3 Mine and Send Funds . . . 17
8.3.1 Mine Blocks . . . 17
8.3.2 Check Your Bitcoin Balance . . . 17
8.3.3 Send Funds to Nodes . . . 18
8.3.4 Verify Transactions . . . 19
8.3.5 Mine 6 Blocks to Confirm Transactions . . . 19
8.3.6 Check Wallet Balance for Each Node . . . 19

9 Channel Management 21
9.1 Create Lightning Channels . . . 21
9.2 Open Channels . . . 22
9.3 Check Connections for Each Node . . . 24
9.4 Check Channels for Each Node . . . 25

10 Performing Multihop Payments 26
10.1 Generate Invoice . . . 26
10.2 Pay Invoice . . . 26
10.3 How to Monitor Payment Status . . . 27
10.3.1 Monitor Payment in Real Time . . . 27

11 AMP (Atomic Multipath Payments)? 27
11.1 How AMP Works . . . 28
11.2 Steps to Use AMP . . . 28
11.2.1 Enable AMP Support . . . 28
11.2.2 Generate an AMP Invoice . . . 28
11.2.3 Pay an AMP Invoice . . . 28
11.2.4 AMP status . . . 29

12 List Payments in the Lightning Network 29

13 Closing Channels 30
13.1 Close Channel for Nodes . . . 30
13.2 Finalize Closure . . . 31

14 Conclusion 31

Acknowledgment

*I would like to express my deepest gratitude to **Dr. Anisur Rahaman Molla**, whose expert guidance, encouragement, and invaluable insights have been a constant source of motivation throughout this project. His mentorship has significantly contributed to the successful completion of my work, and I am immensely thankful for his support. I would also like to extend my heartfelt thanks to **Subhra Mazumdar** for their continuous guidance, assistance, and encouragement during the course of this project. Their expertise and valuable suggestions have greatly enhanced the quality of my work. Finally, I am grateful to my friends and colleagues for their unwavering support and encouragement.*

1 Introduction

This document provides step-by-step guidance on setting up Bitcoin Core in `regtest` mode and configuring multiple Lightning Network Daemon (LND) nodes for testing and development purposes.

2 Differences Between Mainnet, Testnet, and Regtest

Bitcoin networks provide three distinct environments: **Mainnet**, **Testnet**, and **Regtest**. Each serves unique purposes, ranging from real-world transactions to testing and development. Below is a comparison:

Feature	**Mainnet**	**Testnet**	**Regtest**
Purpose	Real Bitcoin transactions with actual value.	Public testing network for developers and testers.	Private testing environment for local development.
Real Money	Yes, Bitcoin on Mainnet has real value.	No, Testnet Bitcoin is valueless.	No, Bitcoin is valueless and locally generated.
Blockchain	Uses the actual Bitcoin blockchain.	Uses a separate public blockchain.	Creates a local blockchain on your computer.
Mining Difficulty	Normal difficulty adjusted globally.	Lower difficulty to simplify testing.	Very low difficulty; blocks can be mined instantly.
Access	Global and decentralized, used by everyone.	Public, open to anyone for testing.	Private, controlled locally by the user.
Faucet Needed	Real Bitcoin is purchased through exchanges or mining.	Testnet Bitcoin is obtained via free faucets.	No faucet needed; you can mine test Bitcoin locally.
Block Explorer	Transactions visible on mainnet explorers (e.g., `blockchain.com`).	Transactions visible on testnet explorers (e.g., `testnet.blockchain.info`).	No public explorer; transactions are visible locally.
Risk	Real economic impact; mistakes can result in monetary loss.	No real value; safe for testing but public.	Completely private and safe for development.
Use Case	Sending and receiving real Bitcoin for real-world applications.	Testing wallets, smart contracts, and other applications.	Debugging and developing applications in a controlled environment.

When to Use Mainnet, Testnet, and Regtest

- **Mainnet:** Use when conducting real Bitcoin transactions with monetary value.

- **Testnet:** Use when testing applications, wallets, or smart contracts in a public network without risks.
- **Regtest:** Use when developing locally, requiring full control and rapid experimentation.

3 Installing Bitcoin Core

3.1 Download and Extract Bitcoin Core

Execute the following commands to download and extract Bitcoin Core:

```
wget https://bitcoincore.org/bin/bitcoin-core-27.0/bitcoin-27.0-x86_64-linux-gnu.tar.gz
tar -xvzf bitcoin-27.0-x86_64-linux-gnu.tar.gz
sudo mv bitcoin-27.0/bin/* /usr/local/bin/
```

3.2 Verify Installation

To confirm the installation, run:

```
bitcoind --version
```

4 Configuring Bitcoin Core

4.1 Edit Configuration File

Open the configuration file:

```
nano ~/.bitcoin/bitcoin.conf
```

Add the following lines:

```
regtest=1
server=1
fallbackfee=0.0002
rpcuser=kuddus24
rpcpassword=Kuddus@786
zmqpubrawblock=tcp://127.0.0.1:28332
zmqpubrawtx=tcp://127.0.0.1:28333
```

4.2 Start Bitcoin Core

Run the following command to start Bitcoin Core in `regtest` mode:

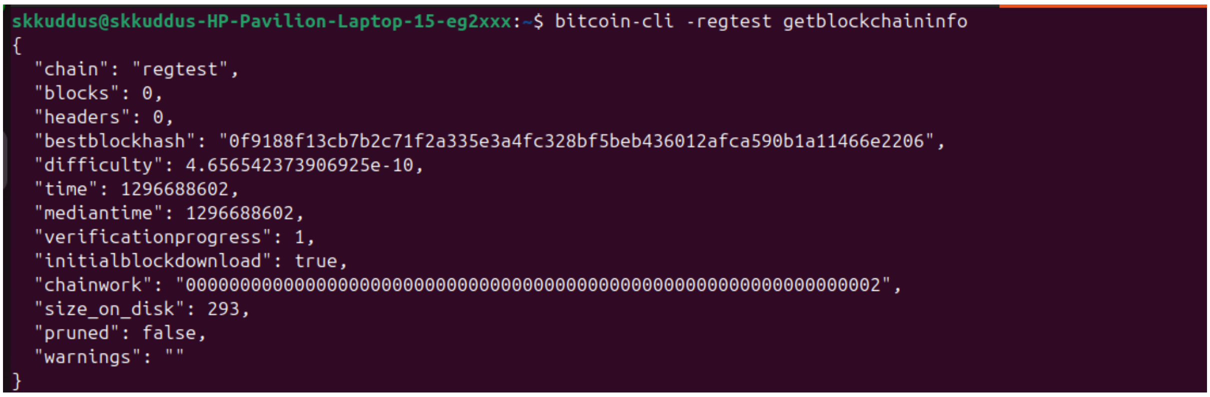
```
bitcoind -regtest -daemon
```

Monitor the logs:

```
tail -f ~/.bitcoin/regtest/debug.log
```

Verify the blockchain information:

```
bitcoin-cli -regtest getblockchaininfo
```



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~$ bitcoin-cli -regtest getblockchaininfo
{
  "chain": "regtest",
  "blocks": 0,
  "headers": 0,
  "bestblockhash": "0f9188f13cb7b2c71f2a335e3a4fc328bf5beb436012afca590b1a11466e2206",
  "difficulty": 4.656542373906925e-10,
  "time": 1296688602,
  "mediantime": 1296688602,
  "verificationprogress": 1,
  "initialblockdownload": true,
  "chainwork": "0000000000000000000000000000000000000000000000000000000000000002",
  "size_on_disk": 293,
  "pruned": false,
  "warnings": ""
}
```

5 Installing Go and LND

5.1 Install Go

Go (Golang) plays a critical role in our project as it serves as the backbone for building and managing the Lightning Network Daemon (LND), which is a key component of the Lightning Network protocol. As LND is written in Go, the language's concurrency model and performance capabilities make it ideal for handling high-speed, low-latency blockchain transactions. Additionally, Go enables the creation of custom tools for automating workflows, managing payment channels, and interacting seamlessly with Bitcoin Core via RPC (*Remote Procedure Call (RPC) is a protocol that allows a program to request services or execute procedures on a remote machine as if they were local. In blockchain systems, RPC serves as a critical communication mechanism between clients, servers, and nodes. For instance, Bitcoin Core provides an RPC interface to interact with the blockchain, enabling tasks such as querying blockchain information, sending transactions, or mining blocks. Similarly, the Lightning Network Daemon (LND) leverages gRPC to manage payment channels, send payments, and automate tasks.*) and ZeroMQ. Its simplicity, efficiency, and cross-platform compatibility further enhance the scalability and reliability of our blockchain-based applications.

Execute the following commands:

```
wget https://dl.google.com/go/go1.22.6.linux-amd64.tar.gz
sudo rm -rf /usr/local/go && sudo tar -C /usr/local -xzf go1.22.6.linux-amd64.tar.gz
export PATH=$PATH:/usr/local/go/bin
export GOPATH=~/go
export PATH=$PATH:$GOPATH/bin
```

5.2 Install LND

Clone the repository and build from source:

```
git clone https://github.com/lightningnetwork/lnd
cd lnd
make install
```

6 Configuring Lightning Nodes

6.1 Set Up Directories

Create directories for the nodes:

```
mkdir -p ~/lnd-A ~/lnd-B ~/lnd-C ~/lnd-D
```

6.1.1 Verify directories

```
    ls ~
```

6.2 Create Configuration Files

6.2.1 Node A Configuration

```
    nano ~/lnd-A/lnd.conf
```

File: ~/lnd-A/lnd.conf

```
[Application Options]
alias=A
listen=127.0.0.1:9735
restlisten=127.0.0.1:8081
debuglevel=info
rpclisten=127.0.0.1:10009

[Bitcoin]
bitcoin.active=1
bitcoin.regtest=1
bitcoin.node=bitcoind

[Bitcoind]
bitcoind.rpcuser=kuddus24
bitcoind.rpcpass=**********
bitcoind.zmqpubrawblock=tcp://127.0.0.1:28332
bitcoind.zmqpubrawtx=tcp://127.0.0.1:28333
bitcoind.rpchost=127.0.0.1
```

Explanation of Key Parameters

- `alias`: Sets the alias for the Lightning node. This is a user-friendly name that is displayed in network explorers and for peer connections.
- `listen`: Specifies the local IP address and port where the Lightning node listens for incoming peer-to-peer connections. The default port for Lightning Network is 9735.

- `restlisten`: Configures the REST API server's IP and port. REST APIs allow programmatic interaction with the node via HTTP requests.
- `debuglevel`: Sets the logging level to `info`. Other levels include `debug`, `warn`, `error`, etc., to control verbosity in logs.
- `rpclisten`: Defines the IP and port for gRPC-based communication with the node. gRPC is used by applications interfacing with the Lightning node.

[**Bitcoin**]

- `bitcoin.active`: Activates the Bitcoin backend for the Lightning node, ensuring it works on the Bitcoin network.
- `bitcoin.regtest`: Enables *regtest mode* (Regression Testing Network) for Bitcoin. This mode is used for testing and development, allowing the creation of a private Bitcoin network with faster block generation.
- `bitcoin.node`: Specifies the backend Bitcoin node to use. In this case, it uses `bitcoind` (the standard Bitcoin Core node).

[**Bitcoind**]

- `bitcoind.rpcuser`: The username for authenticating with the Bitcoin Core node's RPC interface. It must match the `rpcuser` set in the Bitcoin Core configuration file (`bitcoin.conf`).
- `bitcoind.rpcpass`: The password for authenticating with the Bitcoin Core node's RPC interface. It is important to keep this password secure and private.
- `bitcoind.zmqpubrawblock`: Specifies the ZeroMQ (ZMQ) endpoint for subscribing to raw block notifications. ZMQ provides real-time updates on new blocks added to the blockchain.
- `bitcoind.zmqpubrawtx`: Specifies the ZeroMQ endpoint for subscribing to raw transaction notifications. This is useful for monitoring real-time transactions.
- `bitcoind.rpchost`: The host address of the Bitcoin Core RPC server. The value `127.0.0.1` means the RPC interface is accessible locally.

Save the File and Exit

After adding the necessary configuration, save the file and exit the editor:

- In `nano`, press `CTRL + X`, then press `Y` to confirm saving, and finally press `Enter`.

LND and Bitcoin Core RPC Communication

LND (Lightning Network Daemon) requires communication with a Bitcoin node to function properly. Specifically, LND tries to communicate with a Bitcoin node running on `localhost` (`127.0.0.1`) at port `18332`. This port is the default for Bitcoin Core's testnet RPC connection.

Creating Lightning Nodes B, C, and D

Repeat for other nodes `B`, `C`, and `D`, updating aliases and ports accordingly.

6.2.2 Node B Configuration

```
    nano ~/lnd-B/lnd.conf
```

File: `~/lnd-B/lnd.conf`

```
[Application Options]
alias=B
listen=127.0.0.1:9736
restlisten=127.0.0.1:8082
debuglevel=info
rpclisten=127.0.0.1:10010

[Bitcoin]
bitcoin.active=1
bitcoin.regtest=1
bitcoin.node=bitcoind

[Bitcoind]
bitcoind.rpcuser=kuddus24
bitcoind.rpcpass=**********
bitcoind.zmqpubrawblock=tcp://127.0.0.1:28332
bitcoind.zmqpubrawtx=tcp://127.0.0.1:28333
bitcoind.rpchost=127.0.0.1
```

Listing 1: Node B Configuration

6.2.3 Node C Configuration

```
    nano ~/lnd-C/lnd.conf
```

File: `~/lnd-C/lnd.conf`

```
[Application Options]
alias=C
listen=127.0.0.1:9737
restlisten=127.0.0.1:8083
debuglevel=info
rpclisten=127.0.0.1:10011

[Bitcoin]
bitcoin.active=1
bitcoin.regtest=1
bitcoin.node=bitcoind

```

```
[Bitcoind]
bitcoind.rpcuser=kuddus24
bitcoind.rpcpass=**********
bitcoind.zmqpubrawblock=tcp://127.0.0.1:28332
bitcoind.zmqpubrawtx=tcp://127.0.0.1:28333
bitcoind.rpchost=127.0.0.1
```

Listing 2: Node B Configuration

6.2.4 Node D Configuration

```
    nano ~/lnd-D/lnd.conf
```

File: ~/lnd-D/lnd.conf

```
[Application Options]
alias=D
listen=127.0.0.1:9738
restlisten=127.0.0.1:8084
debuglevel=info
rpclisten=127.0.0.1:10012

[Bitcoin]
bitcoin.active=1
bitcoin.regtest=1
bitcoin.node=bitcoind

[Bitcoind]
bitcoind.rpcuser=kuddus24
bitcoind.rpcpass=*********
bitcoind.zmqpubrawblock=tcp://127.0.0.1:28332
bitcoind.zmqpubrawtx=tcp://127.0.0.1:28333
bitcoind.rpchost=127.0.0.1
```

Listing 3: Node B Configuration

6.3 Start Each Node

Run the following commands to start the nodes:*Run each command individually and then create wallet for respetive node.Follow this procedure for all four node.*

```
lnd --lnddir=~/lnd-A --configfile=~/lnd-A/lnd.conf
lnd --lnddir=~/lnd-B --configfile=~/lnd-B/lnd.conf
lnd --lnddir=~/lnd-C --configfile=~/lnd-C/lnd.conf
lnd --lnddir=~/lnd-D --configfile=~/lnd-D/lnd.conf
```

7 Wallet Setup and Funding

Step-by-Step: Create Wallets for Each Node

Node A

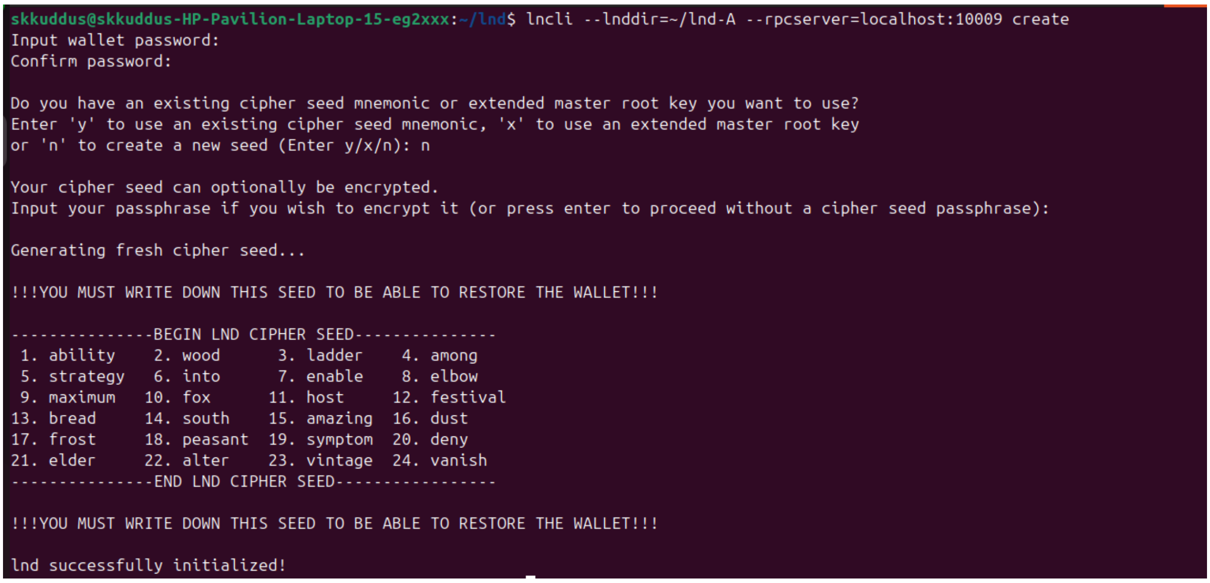
Run the following command:

```
lncli --lnddir=~/lnd-A --rpcserver=localhost:10009 create
```

Listing 4: Create Wallet for Node A

Steps:

- Enter and confirm a wallet password.
- Write down the recovery mnemonic seed when prompted.



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --rpcserver=localhost:10009 create
Input wallet password:
Confirm password:

Do you have an existing cipher seed mnemonic or extended master root key you want to use?
Enter 'y' to use an existing cipher seed mnemonic, 'x' to use an extended master root key
or 'n' to create a new seed (Enter y/x/n): n

Your cipher seed can optionally be encrypted.
Input your passphrase if you wish to encrypt it (or press enter to proceed without a cipher seed passphrase):

Generating fresh cipher seed...

!!!YOU MUST WRITE DOWN THIS SEED TO BE ABLE TO RESTORE THE WALLET!!!

---------------BEGIN LND CIPHER SEED---------------
 1. ability    2. wood       3. ladder     4. among
 5. strategy   6. into       7. enable     8. elbow
 9. maximum   10. fox       11. host      12. festival
13. bread     14. south     15. amazing   16. dust
17. frost     18. peasant   19. symptom   20. deny
21. elder     22. alter     23. vintage   24. vanish
---------------END LND CIPHER SEED-----------------

!!!YOU MUST WRITE DOWN THIS SEED TO BE ABLE TO RESTORE THE WALLET!!!

lnd successfully initialized!
```

Node B

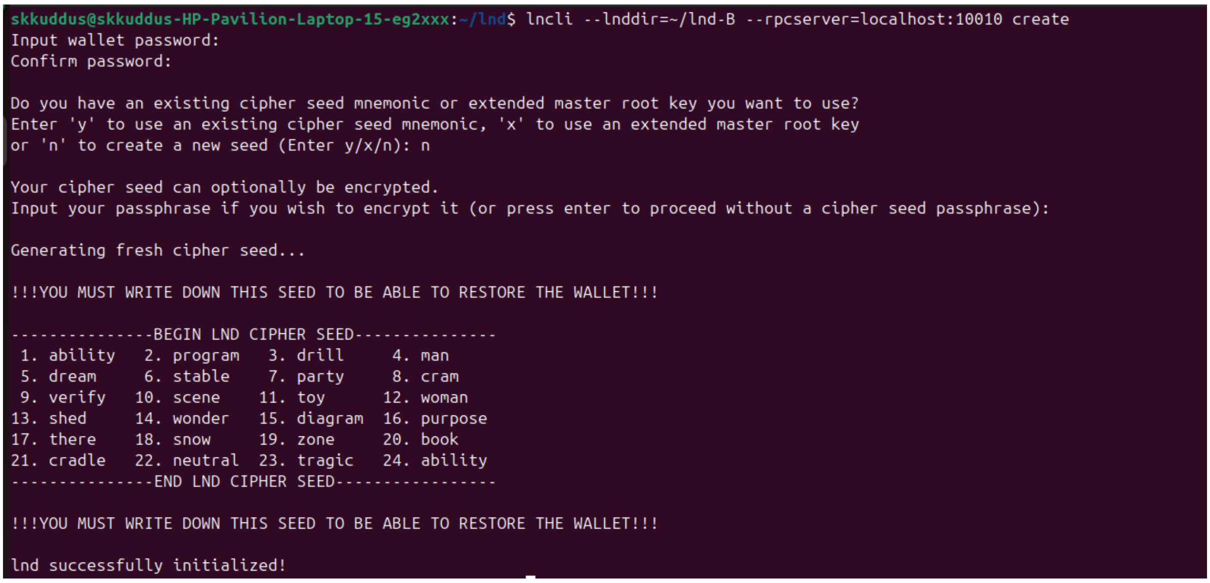
Run the following command:

```
lncli --lnddir=~/lnd-B --rpcserver=localhost:10010 create
```

Listing 5: Create Wallet for Node B

Steps:

- Enter and confirm a wallet password.
- Write down the recovery mnemonic seed when prompted.



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-B --rpcserver=localhost:10010 create
Input wallet password:
Confirm password:

Do you have an existing cipher seed mnemonic or extended master root key you want to use?
Enter 'y' to use an existing cipher seed mnemonic, 'x' to use an extended master root key
or 'n' to create a new seed (Enter y/x/n): n

Your cipher seed can optionally be encrypted.
Input your passphrase if you wish to encrypt it (or press enter to proceed without a cipher seed passphrase):

Generating fresh cipher seed...

!!!YOU MUST WRITE DOWN THIS SEED TO BE ABLE TO RESTORE THE WALLET!!!

---------------BEGIN LND CIPHER SEED---------------
 1. ability   2. program   3. drill     4. man
 5. dream     6. stable    7. party     8. cram
 9. verify   10. scene    11. toy      12. woman
13. shed     14. wonder   15. diagram  16. purpose
17. there    18. snow     19. zone     20. book
21. cradle   22. neutral  23. tragic   24. ability
---------------END LND CIPHER SEED-----------------

!!!YOU MUST WRITE DOWN THIS SEED TO BE ABLE TO RESTORE THE WALLET!!!

lnd successfully initialized!
```

Node C

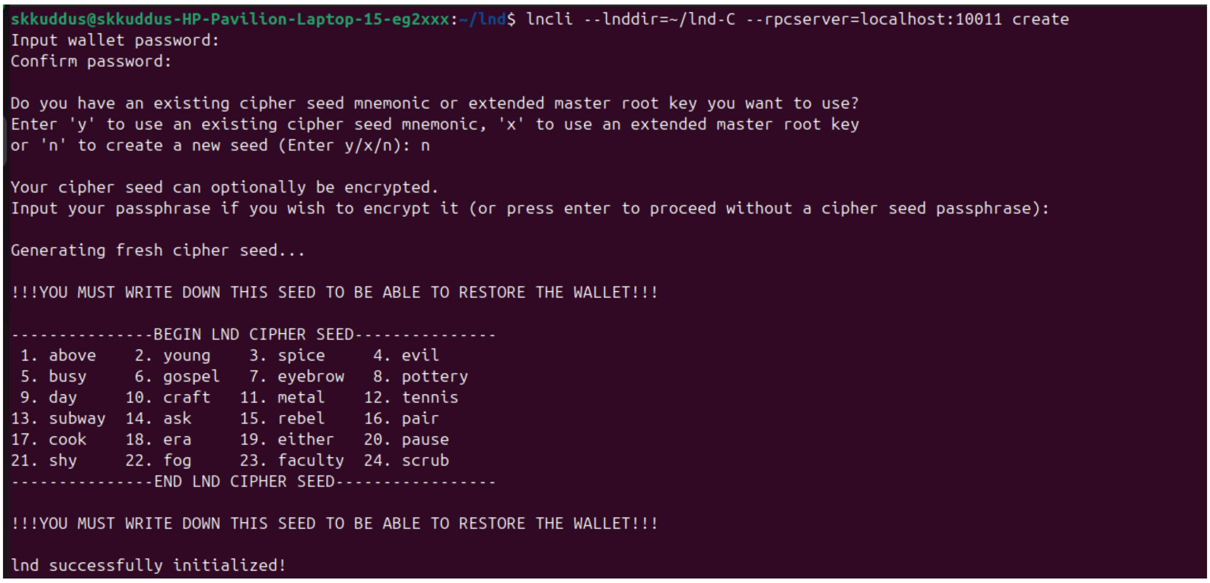
Run the following command:

```
lncli --lnddir=~/lnd-C --rpcserver=localhost:10011 create
```

Listing 6: Create Wallet for Node C

Steps:

- Enter and confirm a wallet password.
- Write down the recovery mnemonic seed when prompted.



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-C --rpcserver=localhost:10011 create
Input wallet password:
Confirm password:

Do you have an existing cipher seed mnemonic or extended master root key you want to use?
Enter 'y' to use an existing cipher seed mnemonic, 'x' to use an extended master root key
or 'n' to create a new seed (Enter y/x/n): n

Your cipher seed can optionally be encrypted.
Input your passphrase if you wish to encrypt it (or press enter to proceed without a cipher seed passphrase):

Generating fresh cipher seed...

!!!YOU MUST WRITE DOWN THIS SEED TO BE ABLE TO RESTORE THE WALLET!!!

---------------BEGIN LND CIPHER SEED---------------
 1. above    2. young    3. spice     4. evil
 5. busy     6. gospel   7. eyebrow   8. pottery
 9. day     10. craft   11. metal    12. tennis
13. subway  14. ask     15. rebel    16. pair
17. cook    18. era     19. either   20. pause
21. shy     22. fog     23. faculty  24. scrub
---------------END LND CIPHER SEED-----------------

!!!YOU MUST WRITE DOWN THIS SEED TO BE ABLE TO RESTORE THE WALLET!!!

lnd successfully initialized!
```

Node D

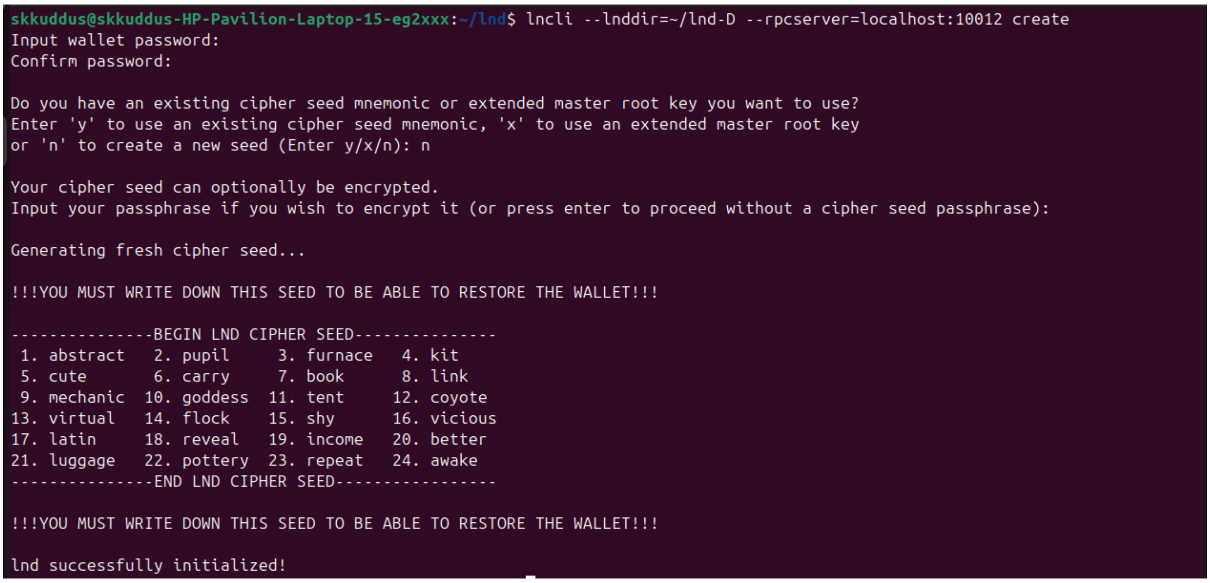
Run the following command:

```
lncli --lnddir=~/lnd-D --rpcserver=localhost:10012 create
```

Listing 7: Create Wallet for Node D

Steps:

- Enter and confirm a wallet password.
- Write down the recovery mnemonic seed when prompted.



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-D --rpcserver=localhost:10012 create
Input wallet password:
Confirm password:

Do you have an existing cipher seed mnemonic or extended master root key you want to use?
Enter 'y' to use an existing cipher seed mnemonic, 'x' to use an extended master root key
or 'n' to create a new seed (Enter y/x/n): n

Your cipher seed can optionally be encrypted.
Input your passphrase if you wish to encrypt it (or press enter to proceed without a cipher seed passphrase):

Generating fresh cipher seed...

!!!YOU MUST WRITE DOWN THIS SEED TO BE ABLE TO RESTORE THE WALLET!!!

---------------BEGIN LND CIPHER SEED---------------
 1. abstract   2. pupil     3. furnace   4. kit
 5. cute       6. carry     7. book      8. link
 9. mechanic  10. goddess  11. tent     12. coyote
13. virtual   14. flock    15. shy      16. vicious
17. latin     18. reveal   19. income   20. better
21. luggage   22. pottery  23. repeat   24. awake
---------------END LND CIPHER SEED-----------------

!!!YOU MUST WRITE DOWN THIS SEED TO BE ABLE TO RESTORE THE WALLET!!!

lnd successfully initialized!
```

7.1 Unlock Wallets

If the nodes are restarted, we'll need to unlock each wallet individually using:

```
lncli --lnddir=~/lnd-A --rpcserver=localhost:10009 unlock
lncli --lnddir=~/lnd-B --rpcserver=localhost:10010 unlock
lncli --lnddir=~/lnd-C --rpcserver=localhost:10011 unlock
lncli --lnddir=~/lnd-D --rpcserver=localhost:10012 unlock
```

Listing 8: Unlock Wallets

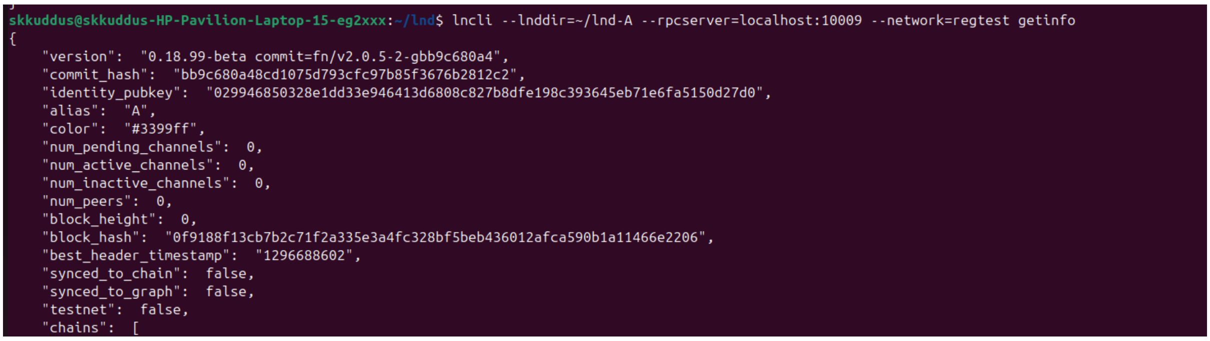
7.2 Retrieve Public Key and Node Information

Run the following commands to retrieve the public key and other details for each Lightning Network node.

Node A

```
lncli --lnddir=~/lnd-A --rpcserver=localhost:10009 --network=
    regtest getinfo
```

Listing 9: Command for Node A

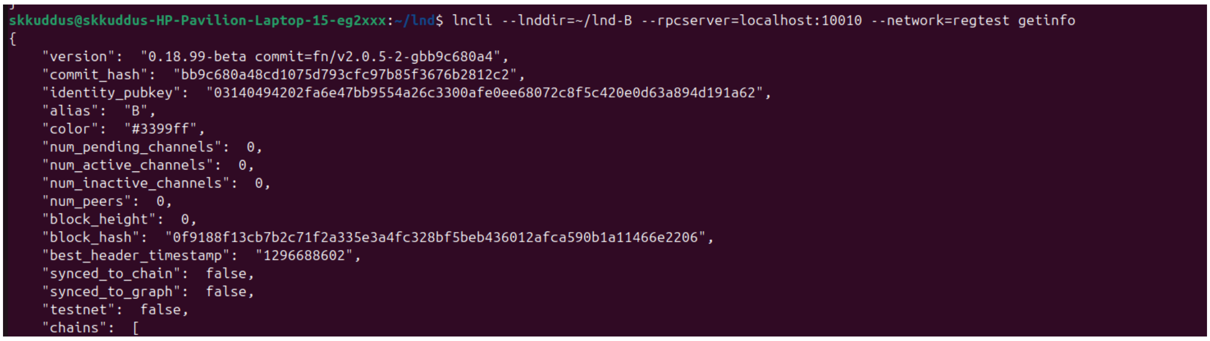


```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --rpcserver=localhost:10009 --network=regtest getinfo
{
    "version":  "0.18.99-beta commit=fn/v2.0.5-2-gbb9c680a4",
    "commit_hash":  "bb9c680a48cd1075d793cfc97b85f3676b2812c2",
    "identity_pubkey":  "029946850328e1dd33e946413d6808c827b8dfe198c393645eb71e6fa5150d27d0",
    "alias":  "A",
    "color":  "#3399ff",
    "num_pending_channels":  0,
    "num_active_channels":  0,
    "num_inactive_channels":  0,
    "num_peers":  0,
    "block_height":  0,
    "block_hash":  "0f9188f13cb7b2c71f2a335e3a4fc328bf5beb436012afca590b1a11466e2206",
    "best_header_timestamp":  "1296688602",
    "synced_to_chain":  false,
    "synced_to_graph":  false,
    "testnet":  false,
    "chains":  [
```

Node B

```
lncli --lnddir=~/lnd-B --rpcserver=localhost:10010 --network=
   regtest getinfo
```

Listing 10: Command for Node B

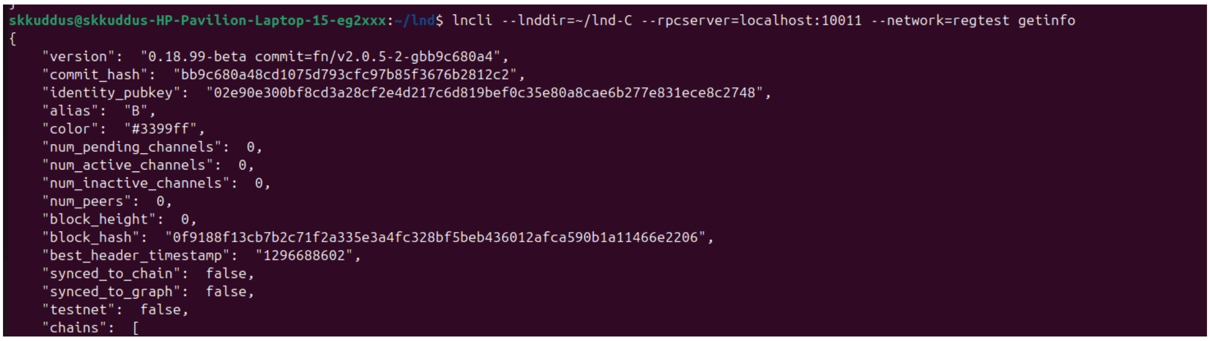


```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-B --rpcserver=localhost:10010 --network=regtest getinfo
{
    "version":  "0.18.99-beta commit=fn/v2.0.5-2-gbb9c680a4",
    "commit_hash":  "bb9c680a48cd1075d793cfc97b85f3676b2812c2",
    "identity_pubkey":  "03140494202fa6e47bb9554a26c3300afe0ee68072c8f5c420e0d63a894d191a62",
    "alias":  "B",
    "color":  "#3399ff",
    "num_pending_channels":  0,
    "num_active_channels":  0,
    "num_inactive_channels":  0,
    "num_peers":  0,
    "block_height":  0,
    "block_hash":  "0f9188f13cb7b2c71f2a335e3a4fc328bf5beb436012afca590b1a11466e2206",
    "best_header_timestamp":  "1296688602",
    "synced_to_chain":  false,
    "synced_to_graph":  false,
    "testnet":  false,
    "chains":  [
```

Node C

```
lncli --lnddir=~/lnd-C --rpcserver=localhost:10011 --network=
   regtest getinfo
```

Listing 11: Command for Node C

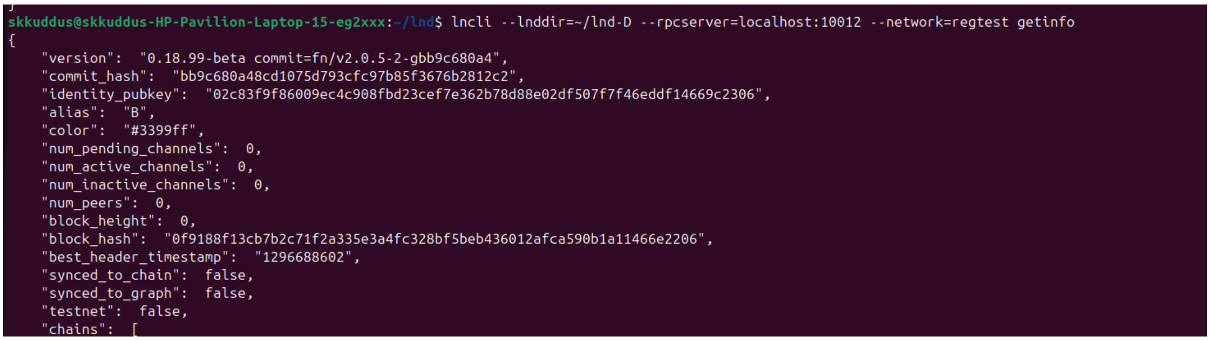


```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-C --rpcserver=localhost:10011 --network=regtest getinfo
{
    "version":  "0.18.99-beta commit=fn/v2.0.5-2-gbb9c680a4",
    "commit_hash":  "bb9c680a48cd1075d793cfc97b85f3676b2812c2",
    "identity_pubkey":  "02e90e300bf8cd3a28cf2e4d217c6d819bef0c35e80a8cae6b277e831ece8c2748",
    "alias":  "B",
    "color":  "#3399ff",
    "num_pending_channels":  0,
    "num_active_channels":  0,
    "num_inactive_channels":  0,
    "num_peers":  0,
    "block_height":  0,
    "block_hash":  "0f9188f13cb7b2c71f2a335e3a4fc328bf5beb436012afca590b1a11466e2206",
    "best_header_timestamp":  "1296688602",
    "synced_to_chain":  false,
    "synced_to_graph":  false,
    "testnet":  false,
    "chains":  [
```

Node D

```
lncli --lnddir=~/lnd-D --rpcserver=localhost:10012 --network=
   regtest getinfo
```

Listing 12: Command for Node D



```
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-D --rpcserver=localhost:10012 --network=regtest getinfo
{
    "version":  "0.18.99-beta commit=fn/v2.0.5-2-gbb9c680a4",
    "commit_hash":  "bb9c680a48cd1075d793cfc97b85f3676b2812c2",
    "identity_pubkey":  "02c83f9f86009ec4c908fbd23cef7e362b78d88e02df507f7f46eddf14669c2306",
    "alias":  "B",
    "color":  "#3399ff",
    "num_pending_channels":  0,
    "num_active_channels":  0,
    "num_inactive_channels":  0,
    "num_peers":  0,
    "block_height":  0,
    "block_hash":  "0f9188f13cb7b2c71f2a335e3a4fc328bf5beb436012afca590b1a11466e2206",
    "best_header_timestamp":  "1296688602",
    "synced_to_chain":  false,
    "synced_to_graph":  false,
    "testnet":  false,
    "chains":  [
```

7.3 Generate a New Wallet Address for Each Node

To enable on-chain funding of Lightning channels and to receive on-chain Bitcoin transactions, we need to generate a wallet address for each Lightning node. Below are the commands for generating a new wallet address for Nodes A, B, C, and D.

Node A

Run the following command to generate a new wallet address for Node A:

```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
   :10009 newaddress p2wkh
```

Listing 13: Generate Wallet Address for Node A

Node B

Run the following command to generate a new wallet address for Node B:

```
lncli --lnddir=~/lnd-B --network=regtest --rpcserver=localhost
   :10010 newaddress p2wkh
```

Listing 14: Generate Wallet Address for Node B

Node C

Run the following command to generate a new wallet address for Node C:

```
lncli --lnddir=~/lnd-C --network=regtest --rpcserver=localhost
   :10011 newaddress p2wkh
```

Listing 15: Generate Wallet Address for Node C

Node D

Run the following command to generate a new wallet address for Node D:

```
lncli --lnddir=~/lnd-D --network=regtest --rpcserver=localhost
   :10012 newaddress p2wkh
```

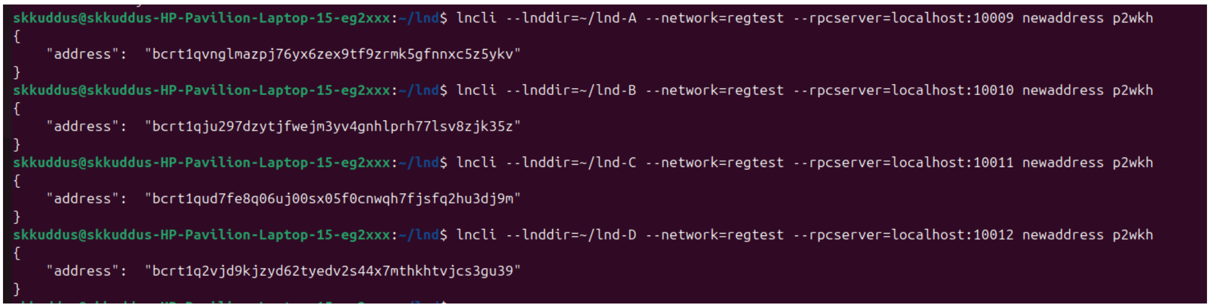
Listing 16: Generate Wallet Address for Node D

Why Do We Need to Generate Wallet Addresses?

- **Funding Lightning Channels**: A wallet address is used to receive Bitcoin on-chain. These funds can be used to open Lightning channels, which are essential for conducting Lightning Network transactions.
- **Receiving On-Chain Payments**: If someone needs to send Bitcoin to our Lightning node directly on-chain, the wallet address provides a destination for those funds.
- **Testing in Regtest Environment**: On the `regtest` network, you can generate test Bitcoin and send it to the wallet address for experimentation and development purposes.

How to Use the Generated Wallet Address?

Once the command is executed, it will output a new wallet address in the following format:



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 newaddress p2wkh
{
    "address":  "bcrt1qvnglmazpj76yx6zex9tf9zrmk5gfnnxc5z5ykv"
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-B --network=regtest --rpcserver=localhost:10010 newaddress p2wkh
{
    "address":  "bcrt1qju297dzytjfwejm3yv4gnhlprh77lsv8zjk35z"
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-C --network=regtest --rpcserver=localhost:10011 newaddress p2wkh
{
    "address":  "bcrt1qud7fe8q06uj00sx05f0cnwqh7fjsfq2hu3dj9m"
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-D --network=regtest --rpcserver=localhost:10012 newaddress p2wkh
{
    "address":  "bcrt1q2vjd9kjzyd62tyedv2s44x7mthkhtvjcs3gu39"
}
```

We can use this address in your Bitcoin Core node or another wallet to send funds to our Lightning node. These funds will be reflected in the node's on-chain wallet balance.

8 Bitcoin Commands and Explanations

8.1 Create a New Wallet

To create a new wallet named `Alice`, run the command:

```
bitcoin-cli -named createwallet wallet_name="My_Wallet"
```

Listing 17: Create a New Wallet

Reason: This command initializes a new wallet called `"My_Wallet"`. Separate wallets help organize funds for different purposes or users, enabling better management and security.

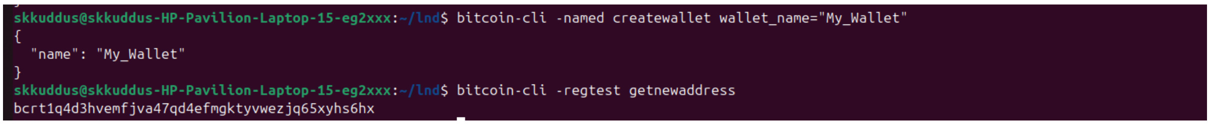
8.2 Generate a New Bitcoin Address

To generate a new Bitcoin address in the `regtest` environment, use the following command:

```
bitcoin-cli -regtest getnewaddress
```

Listing 18: Generate a New Bitcoin Address

Reason: This command creates a new Bitcoin address in the local `regtest` network. The address can be used to receive Bitcoin transactions for testing or funding purposes.



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -named createwallet wallet_name="My_Wallet"
{
  "name": "My_Wallet"
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -regtest getnewaddress
bcrt1q4d3hvemfjva47qd4efmgktyvwezjq65xyhs6hx
```

8.3 Mine and Send Funds

8.3.1 Mine Blocks

Use the following command to mine blocks and send the rewards to a specified address:

```
bitcoin-cli -regtest generatetoaddress 200
    bcrt1q4d3hvemfjva47qd4efmgktyvwezjq65xyhs6hx
```

Listing 19: Mine Blocks

Explanation: This command mines 200 blocks in the `regtest` environment and sends the block rewards to the specified address (`bcrt1q4d3hvemfjva47qd4e fmgktyvwezjq65xyhs6hx`). Mining in `regtest` generates test Bitcoin, which can be used to fund Lightning channels or for other testing purposes.

200: The number of blocks to mine.

bcrt1...: The Bitcoin address to receive the mining rewards.

note: The reward per block starts at 50 BTC and halves approximately every 210,000 blocks

8.3.2 Check Your Bitcoin Balance

To check the current balance of your Bitcoin wallet in the `regtest` environment, use the following command:

```
bitcoin-cli -regtest getbalance
```

Listing 20: Check Wallet Balance

Or

```
bitcoin-cli -rpcwallet=My_Wallet getbalance
```

Listing 21: Check Wallet Balance

Explanation: This command retrieves the total balance available in the currently loaded Bitcoin wallet `"Alice"`.It only includes funds that have been confirmed in mined blocks.

Output:

```
5000.00000000
```

The output shows the balance in Bitcoin (BTC). the wallet has 5000 BTC.

Why Check Your Balance?

Ensure your wallet has sufficient funds for transactions, such as opening Lightning channels or sending Bitcoin to other addresses. Verify that mining rewards or received transactions have been successfully credited.

8.3.3 Send Funds to Nodes

Distribute 20 BTC to Each Node

Use the following commands to send 20 BTC to each of the four nodes.

Send to Node A

```
bitcoin-cli -regtest sendtoaddress
   bcrt1qvnglmazpj76yx6zex9tf9zrmk5gfnnxc5z5ykv 20
```

Listing 22: Send 20 BTC to Node A

Explanation: Sends 20 BTC to Node A's wallet address (`bcrt1qvnglmazpj76yx6zex9tf9zrmk5gfnnxc5z5ykv`). After running above command the transaction ID (txid) returned by the command is :

```
6a107f9d4e21dc0827ba6e9fcd2c033a90eda66b241cda5ed140b924a58c09a9
```

Send to Node B

```
bitcoin-cli -regtest sendtoaddress
   bcrt1qju297dzytjfwejm3yv4gnhlprh77lsv8zjk35z 20
```

Listing 23: Send 20 BTC to Node B

Explanation: Sends 20 BTC to Node B's wallet address (`bcrt1qju297dzytjfwejm3yv4gnhlprh77lsv8zjk35z`). After running above command the transaction ID (txid) returned by the command is :

```
2f351dea9cd9d40d7692c83a3f6c499bcfa12cb4ccf2a5726721caeec8b68cfa
```

Send to Node C

```
bitcoin-cli -regtest sendtoaddress
   bcrt1qud7fe8q06uj00sx05f0cnwqh7fjsfq2hu3dj9m 20
```

Listing 24: Send 20 BTC to Node C

Explanation: Sends 20 BTC to Node C's wallet address (`bcrt1qud7fe8q06uj00sx05f0cnwqh7fjsfq2hu3dj9m`). After running above command the transaction ID (txid) returned by the command is :

```
12fe9680e8ef9d0f076aff1e9bceae1430608cacd1aa123d90325b54c35d6c97
```

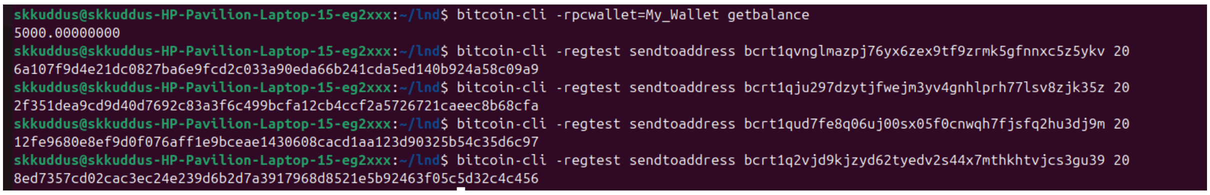
Send to Node D

```
bitcoin-cli -regtest sendtoaddress
   bcrt1q2vjd9kjzyd62tyedv2s44x7mthkhtvjcs3gu39 20
```

Listing 25: Send 20 BTC to Node D

Explanation: Sends 20 BTC to Node D's wallet address (`bcrt1q2vjd9kjzyd62tyedv2s44x7mthkhtvjcs3gu39`). After running above command the transaction ID (txid) returned by the command is :

```
8ed7357cd02cac3ec24e239d6b2d7a3917968d8521e5b92463f05c5d32c4c456
```



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -rpcwallet=My_Wallet getbalance
5000.00000000
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -regtest sendtoaddress bcrt1qvnglmazpj76yx6zex9tf9zrmk5gfnnxc5z5ykv 20
6a107f9d4e21dc0827ba6e9fcd2c033a90eda66b241cda5ed140b924a58c09a9
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -regtest sendtoaddress bcrt1qju297dzytjfwejm3yv4gnhlprh77lsv8zjk35z 20
2f351dea9cd9d40d7692c83a3f6c499bcfa12cb4ccf2a5726721caeec8b68cfa
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -regtest sendtoaddress bcrt1qud7fe8q06uj00sx05f0cnwqh7fjsfq2hu3dj9m 20
12fe9680e8ef9d0f076aff1e9bceae1430608cacd1aa123d90325b54c35d6c97
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -regtest sendtoaddress bcrt1q2vjd9kjzyd62tyedv2s44x7mthkhtvjcs3gu39 20
8ed7357cd02cac3ec24e239d6b2d7a3917968d8521e5b92463f05c5d32c4c456
```

8.3.4 Verify Transactions

After sending the funds, verify the balance of each node using their respective commands. You can also use the following command to check the remaining balance in your wallet:

```
bitcoin-cli -regtest getbalance
```

Listing 26: Check Remaining Wallet Balance

Example Output:

```
4919.99988720
```

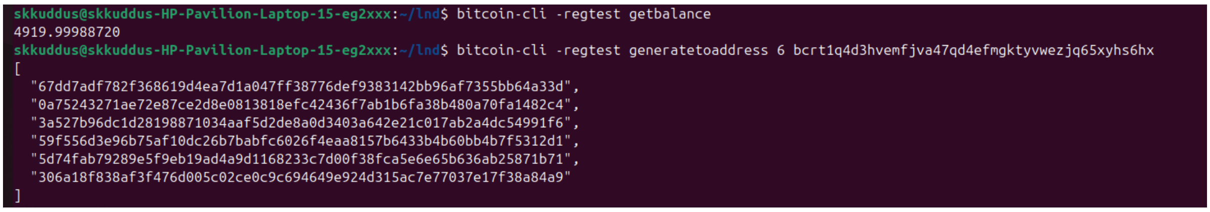
This shows the remaining balance after sending 40 BTC (10 BTC each to 4 nodes) from the mined 50 BTC.

8.3.5 Mine 6 Blocks to Confirm Transactions

To mine 6 blocks in the `regtest` environment, use the following command:

```
bitcoin-cli -regtest generatetoaddress 6
    bcrt1q4d3hvemfjva47qd4efmgktyvwezjq65xyhs6hx
```

Listing 27: Mine 6 Blocks



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -regtest getbalance
4919.99988720
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -regtest generatetoaddress 6 bcrt1q4d3hvemfjva47qd4efmgktyvwezjq65xyhs6hx
[
  "67dd7adf782f368619d4ea7d1a047ff38776def9383142bb96af7355bb64a33d",
  "0a75243271ae72e87ce2d8e0813818efc42436f7ab1b6fa38b480a70fa1482c4",
  "3a527b96dc1d28198871034aaf5d2de8a0d3403a642e21c017ab2a4dc54991f6",
  "59f556d3e96b75af10dc26b7babfc6026f4eaa8157b6433b4b60bb4b7f5312d1",
  "5d74fab79289e5f9eb19ad4a9d1168233c7d00f38fca5e6e65b636ab25871b71",
  "306a18f838af3f476d005c02ce0c9c694649e924d315ac7e77037e17f38a84a9"
]
```

Reason:

- In the Bitcoin mainnet, 6 blocks (1 hour) is considered the threshold for "irreversible" transactions.
- Each additional block reduces the likelihood of a double-spend attack.
- In the `regtest` environment, this process mimics the behavior of the Bitcoin mainnet, where 6 confirmations are standard for transaction finality.Even though a transaction is confirmed in regtest with just 1 block.

8.3.6 Check Wallet Balance for Each Node

Use the following commands to check the wallet balance of each Lightning Network node.

Node A

```
lncli --lnddir=~/lnd-A --rpcserver=localhost:10009 --network=regtest walletbalance
```

Listing 28: Check Balance for Node A

Node B

```
lncli --lnddir=~/lnd-B --rpcserver=localhost:10010 --network=regtest walletbalance
```

Listing 29: Check Balance for Node B

Node C

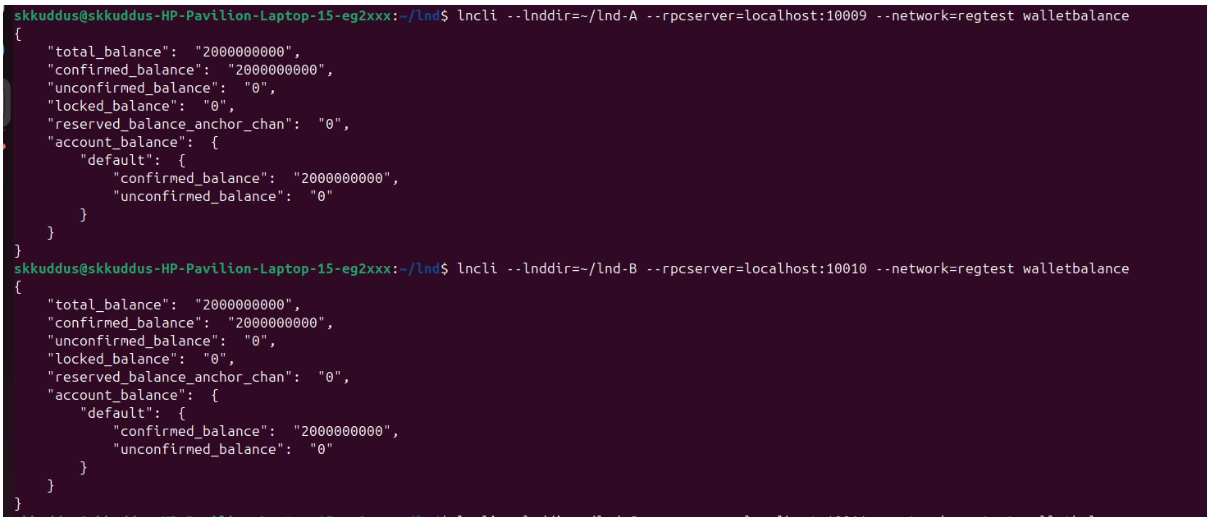
```
lncli --lnddir=~/lnd-C --rpcserver=localhost:10011 --network=regtest walletbalance
```

Listing 30: Check Balance for Node C

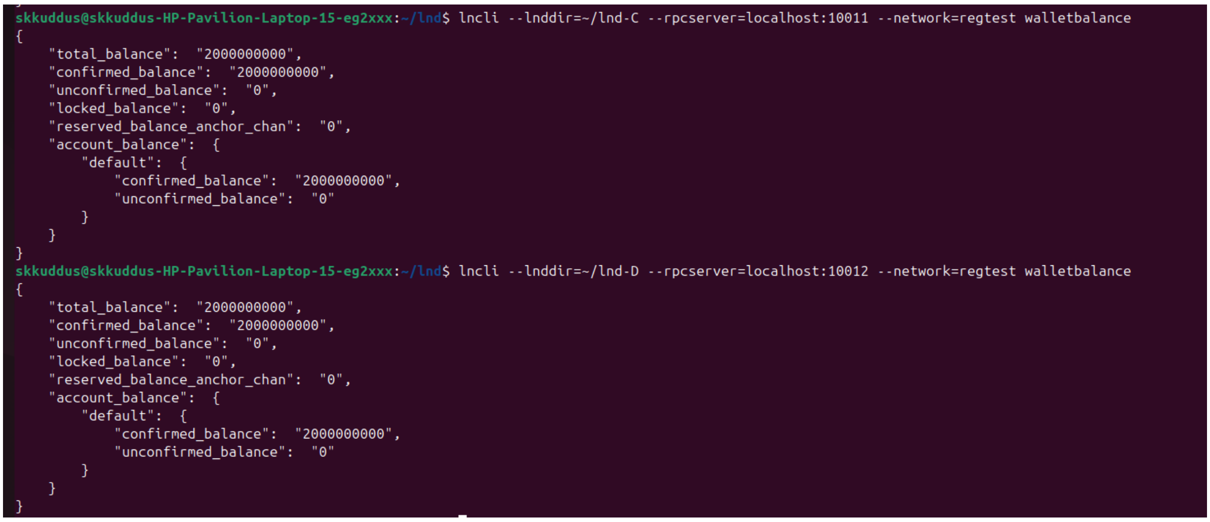
Node D

```
lncli --lnddir=~/lnd-D --rpcserver=localhost:10012 --network=regtest walletbalance
```

Listing 31: Check Balance for Node D



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --rpcserver=localhost:10009 --network=regtest walletbalance
{
    "total_balance":  "2000000000",
    "confirmed_balance":  "2000000000",
    "unconfirmed_balance":  "0",
    "locked_balance":  "0",
    "reserved_balance_anchor_chan":  "0",
    "account_balance":  {
        "default":  {
            "confirmed_balance":  "2000000000",
            "unconfirmed_balance":  "0"
        }
    }
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-B --rpcserver=localhost:10010 --network=regtest walletbalance
{
    "total_balance":  "2000000000",
    "confirmed_balance":  "2000000000",
    "unconfirmed_balance":  "0",
    "locked_balance":  "0",
    "reserved_balance_anchor_chan":  "0",
    "account_balance":  {
        "default":  {
            "confirmed_balance":  "2000000000",
            "unconfirmed_balance":  "0"
        }
    }
}
```



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-C --rpcserver=localhost:10011 --network=regtest walletbalance
{
    "total_balance":  "2000000000",
    "confirmed_balance":  "2000000000",
    "unconfirmed_balance":  "0",
    "locked_balance":  "0",
    "reserved_balance_anchor_chan":  "0",
    "account_balance":  {
        "default":  {
            "confirmed_balance":  "2000000000",
            "unconfirmed_balance":  "0"
        }
    }
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-D --rpcserver=localhost:10012 --network=regtest walletbalance
{
    "total_balance":  "2000000000",
    "confirmed_balance":  "2000000000",
    "unconfirmed_balance":  "0",
    "locked_balance":  "0",
    "reserved_balance_anchor_chan":  "0",
    "account_balance":  {
        "default":  {
            "confirmed_balance":  "2000000000",
            "unconfirmed_balance":  "0"
        }
    }
}
```

Each node's wallet balance reflects the available funds for on-chain transactions and Lightning channel operations.

9 Channel Management

9.1 Create Lightning Channels

To establish peer-to-peer connections between the nodes in the Lightning Network, use the following commands.

Syntax

```
    lncli --lnddir=<node_directory> --network=<network_type> --
       rpcserver=<rpc_server_address> connect <node_pubkey>@<
       node_address>:<node_port>
```

Listing 32: Syntax

Connect Node A to Node B

```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
   :10009 connect 03140494202
   fa6e47bb9554a26c3300afe0ee68072c8f5c420e0d63a894d191a62@127
   .0.0.1:9736
```

Listing 33: Connect Node A to Node B

Connect Node B to Node C

```
lncli --lnddir=~/lnd-B --network=regtest --rpcserver=localhost
   :10010 connect 02
   e90e300bf8cd3a28cf2e4d217c6d819bef0c35e80a8cae6b277e831ece8c2748@127
   .0.0.1:9737
```

Listing 34: Connect Node B to Node C

Connect Node A to Node D

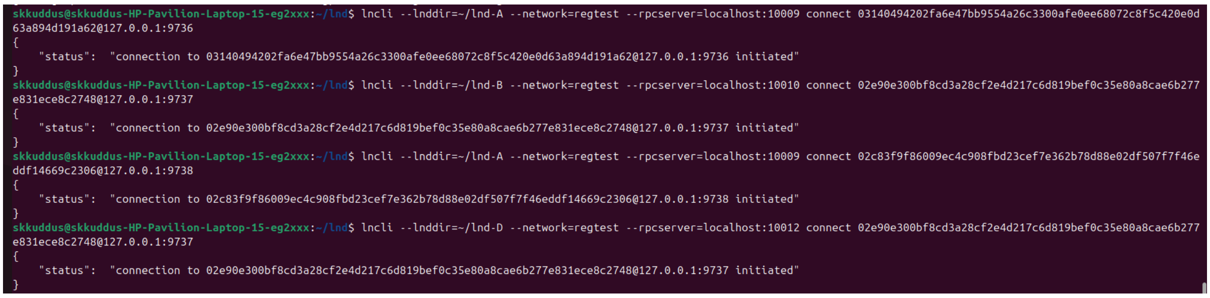
```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
   :10009 connect 02
   c83f9f86009ec4c908fbd23cef7e362b78d88e02df507f7f46eddf14669c2306@127
   .0.0.1:9738
```

Listing 35: Connect Node A to Node D

Connect Node D to Node C

```
lncli --lnddir=~/lnd-D --network=regtest --rpcserver=localhost
   :10012 connect 02
   e90e300bf8cd3a28cf2e4d217c6d819bef0c35e80a8cae6b277e831ece8c2748@127
   .0.0.1:9737
```

Listing 36: Connect Node D to Node C



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 connect 03140494202fa6e47bb9554a26c3300afe0ee68072c8f5c420e0d
63a894d191a62@127.0.0.1:9736
{
    "status":  "connection to 03140494202fa6e47bb9554a26c3300afe0ee68072c8f5c420e0d63a894d191a62@127.0.0.1:9736 initiated"
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-B --network=regtest --rpcserver=localhost:10010 connect 02e90e300bf8cd3a28cf2e4d217c6d819bef0c35e80a8cae6b277
e831ece8c2748@127.0.0.1:9737
{
    "status":  "connection to 02e90e300bf8cd3a28cf2e4d217c6d819bef0c35e80a8cae6b277e831ece8c2748@127.0.0.1:9737 initiated"
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 connect 02c83f9f86009ec4c908fbd23cef7e362b78d88e02df507f7f46e
ddf14669c2306@127.0.0.1:9738
{
    "status":  "connection to 02c83f9f86009ec4c908fbd23cef7e362b78d88e02df507f7f46eddf14669c2306@127.0.0.1:9738 initiated"
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-D --network=regtest --rpcserver=localhost:10012 connect 02e90e300bf8cd3a28cf2e4d217c6d819bef0c35e80a8cae6b277
e831ece8c2748@127.0.0.1:9737
{
    "status":  "connection to 02e90e300bf8cd3a28cf2e4d217c6d819bef0c35e80a8cae6b277e831ece8c2748@127.0.0.1:9737 initiated"
}
```

9.2 Open Channels

Open channels between nodes:

Syntax

```
lncli --lnddir=<node_directory> --network=<network_type> --
   rpcserver=<rpc_server_address> openchannel --node_key=<
   remote_node_pubkey> --local_amt=<amount_in_satoshis>
```

Listing 37: Syntax

From Node A to Node B

```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
   :10009 openchannel --node_key=03140494202
   fa6e47bb9554a26c3300afe0ee68072c8f5c420e0d63a894d191a62 --
   local_amt=100000
```

Listing 38: Open Channel from A to B

From Node B to Node C

```
lncli --lnddir=~/lnd-B --network=regtest --rpcserver=localhost
   :10010 openchannel --node_key=02
   e90e300bf8cd3a28cf2e4d217c6d819bef0c35e80a8cae6b277e831ece8c2748
    --local_amt=100000
```

Listing 39: Open Channel from B to C

From Node A to Node D

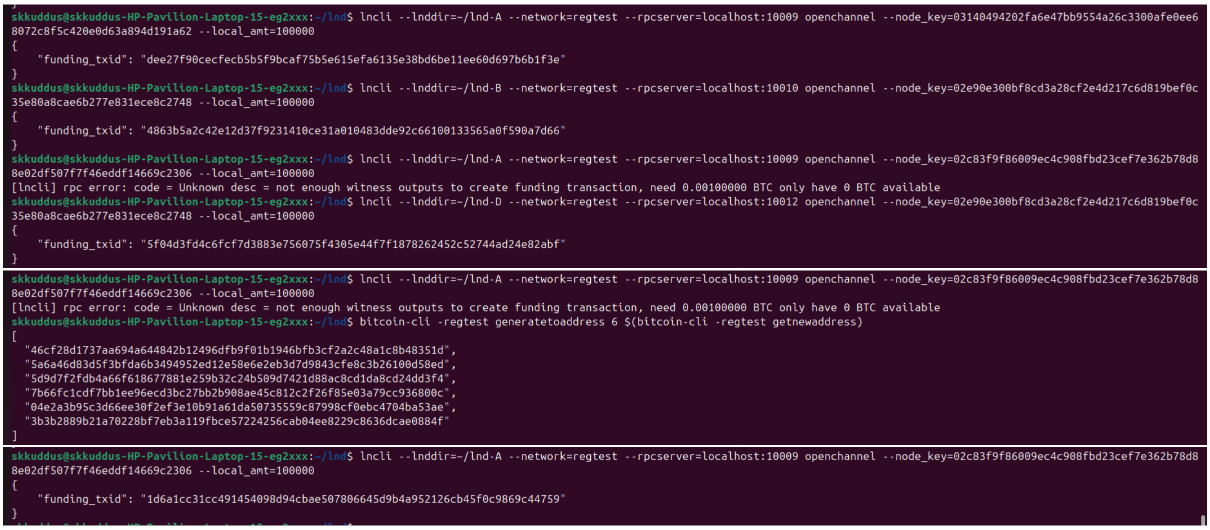
```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
   :10009 openchannel --node_key=02
   c83f9f86009ec4c908fbd23cef7e362b78d88e02df507f7f46eddf14669c2306
    --local_amt=100000
```

Listing 40: Open Channel from A to D

From Node D to Node C

```
lncli --lnddir=~/lnd-D --network=regtest --rpcserver=localhost
   :10012 openchannel --node_key=02
   e90e300bf8cd3a28cf2e4d217c6d819bef0c35e80a8cae6b277e831ece8c2748
    --local_amt=100000
```

Listing 41: Open Channel from D to C



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 openchannel --node_key=03140494202fa6e47bb9554a26c3300afe0ee6
8072c8f5c420e0d63a894d191a62 --local_amt=100000
{
    "funding_txid": "dee27f90cecfecb5b5f9bcaf75b5e615efa6135e38bd6be11ee60d697b6b1f3e"
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-B --network=regtest --rpcserver=localhost:10010 openchannel --node_key=02e90e300bf8cd3a28cf2e4d217c6d819bef0c
35e80a8cae6b277e831ece8c2748 --local_amt=100000
{
    "funding_txid": "4863b5a2c42e12d37f9231410ce31a010483dde92c66100133565a0f590a7d66"
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 openchannel --node_key=02c83f9f86009ec4c908fbd23cef7e362b78d8
8e02df507f7f46eddf14669c2306 --local_amt=100000
[lncli] rpc error: code = Unknown desc = not enough witness outputs to create funding transaction, need 0.00100000 BTC only have 0 BTC available
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-D --network=regtest --rpcserver=localhost:10012 openchannel --node_key=02e90e300bf8cd3a28cf2e4d217c6d819bef0c
35e80a8cae6b277e831ece8c2748 --local_amt=100000
{
    "funding_txid": "5f04d3fd4c6fcf7d3883e756075f4305e44f7f1878262452c52744ad24e82abf"
}
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 openchannel --node_key=02c83f9f86009ec4c908fbd23cef7e362b78d8
8e02df507f7f46eddf14669c2306 --local_amt=100000
[lncli] rpc error: code = Unknown desc = not enough witness outputs to create funding transaction, need 0.00100000 BTC only have 0 BTC available
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -regtest generatetoaddress 6 $(bitcoin-cli -regtest getnewaddress)
[
  "46cf28d1737aa694a644842b12496dfb9f01b1946bfb3cf2a2c48a1c8b48351d",
  "5a6a46d83d5f3bfda6b3494952ed12e58e6e2eb3d7d9843cfe8c3b26100d58ed",
  "5d9d7f2fdb4a66f618677881e259b32c24b509d7421d88ac8cd1da8cd24dd3f4",
  "7b66fc1cdf7bb1ee96ecd3bc27bb2b908ae45c812c2f26f85e03a79cc936800c",
  "04e2a3b95c3d66ee30f2ef3e10b91a61da50735559c87998cf0ebc4704ba53ae",
  "3b3b2889b21a70228bf7eb3a119fbce57224256cab04ee8229c8636dcae0884f"
]
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 openchannel --node_key=02c83f9f86009ec4c908fbd23cef7e362b78d8
8e02df507f7f46eddf14669c2306 --local_amt=100000
{
    "funding_txid": "1d6a1cc31cc491454098d94cbae507806645d9b4a952126cb45f0c9869c44759"
}
```

Steps to Resolve Wallet Sync Issue

Step 1: Check Wallet Sync Status

Before opening a Lightning channel, ensure the wallet is fully synchronized with the blockchain. Run the following command to check the sync status of the node:

```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
    :10009 getinfo
```

Listing 42: Check Wallet Sync Status

Explanation:

- ‘synced_to_chain‘: This field should display `true`, indicating the wallet is synced.
- ‘block_height‘: This field shows the current height of the blockchain as seen by the node.

If the wallet is not synced, proceed to mine additional blocks.

Step 2: Mine Additional Blocks

In `regtest`, transactions require mined blocks for confirmation. Use the following command to mine 10 blocks:

```
bitcoin-cli -regtest generatetoaddress 10 <wallet_address>
```

Listing 43: Mine Additional Blocks

Explanation:

- 10: Number of blocks to mine.
- wallet_address: a valid Bitcoin address from Node A's wallet.

To generate a new wallet address, run:

```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
    :10009 newaddress p2wkh
```

Listing 44: Generate a Wallet Address

Verify the blockchain height using:

```
bitcoin-cli -regtest getblockchaininfo
```

Listing 45: Check Blockchain Height

9.3 Check Connections for Each Node

To check the peer connections for each node in the Lightning Network, use the following commands:

Check Connections for Node A

```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
    :10009 listpeers
```

Listing 46: Node A Peer Connections

Check Connections for Node B

```
lncli --lnddir=˜/lnd-B --network=regtest --rpcserver=localhost
   :10010 listpeers
```

Listing 47: Node B Peer Connections

Check Connections for Node C

```
lncli --lnddir=˜/lnd-C --network=regtest --rpcserver=localhost
   :10011 listpeers
```

Listing 48: Node C Peer Connections

Check Connections for Node D

```
lncli --lnddir=˜/lnd-D --network=regtest --rpcserver=localhost
   :10012 listpeers
```

Listing 49: Node D Peer Connections

9.4 Check Channels for Each Node

To check the peer channels for each node in the Lightning Network, use the following commands:

Check Channels for Node A

```
lncli --lnddir=˜/lnd-A --network=regtest --rpcserver=localhost
   :10009 listchannels
```

Listing 50: Node A Peer Connections

Check Channels for Node B

```
lncli --lnddir=˜/lnd-B --network=regtest --rpcserver=localhost
   :10010 listchannels
```

Listing 51: Node B Peer Connections

Check Channels for Node C

```
lncli --lnddir=˜/lnd-C --network=regtest --rpcserver=localhost
   :10011 listchannels
```

Listing 52: Node C Peer Connections

Check Channels for Node D

```
lncli --lnddir=~/lnd-D --network=regtest --rpcserver=localhost
    :10012 listchannels
```

Listing 53: Node D Peer Connections

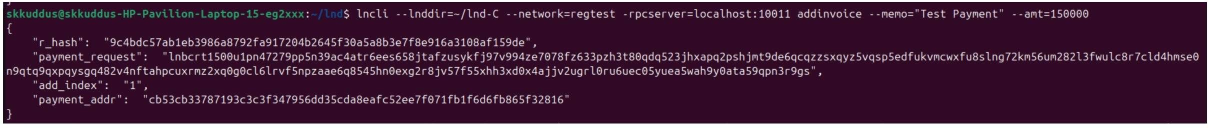
10 Performing Multihop Payments

10.1 Generate Invoice

Generate an invoice on `C`:

```
lncli --lnddir=~/lnd-C --network=regtest -rpcserver=localhost
    :10011 addinvoice --memo="Test Payment" --amt=150000
```

The provided command creates an **invoice** on Node C in the Lightning Network for a payment of `150000 satoshis` (with the memo `"Test Payment"`).



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-C --network=regtest -rpcserver=localhost:10011 addinvoice --memo="Test Payment" --amt=150000
{
    "r_hash":  "9c4bdc57ab1eb3986a8792fa917204b2645f30a5a8b3e7f8e916a3108af159de",
    "payment_request":  "lnbcrt1500u1pn47279pp5n39ac4atr6ees658jtafzusykfj97v994ze7078fz633pzh3t80qdq523jhxapq2pshjmt9de6qcqzzsxqyz5vqsp5edfukvmcwxfu8slng72km56um282l3fwulc8r7cld4hmse0
n9qtq9qxpqysgq482v4nftahpcuxrmz2xq0g0cl6lrvf5npzaae6q8545hn0exg2r8jv57f55xhh3xd0x4ajjv2ugrl0ru6uec05yuea5wah9y0ata59qpn3r9gs",
    "add_index":  "1",
    "payment_addr":  "cb53cb33787193c3c3f347956dd35cda8eafc52ee7f071fb1f6d6fb865f32816"
}
```

What Does This Invoice Do?

- It acts as a **request for payment** that other nodes in the Lightning Network can use to send `150000` to Node C.
- The `payment_request` string (in Bolt11 format) contains all necessary information, including:
  - **Amount**: The amount of the requested payment (`150000` in this case).
  - **Destination Node**: Identifies the receiving node (Node C) in the network.
  - **Payment Hash**: A unique hash to identify and verify the payment.

10.2 Pay Invoice

Pay from `A`:

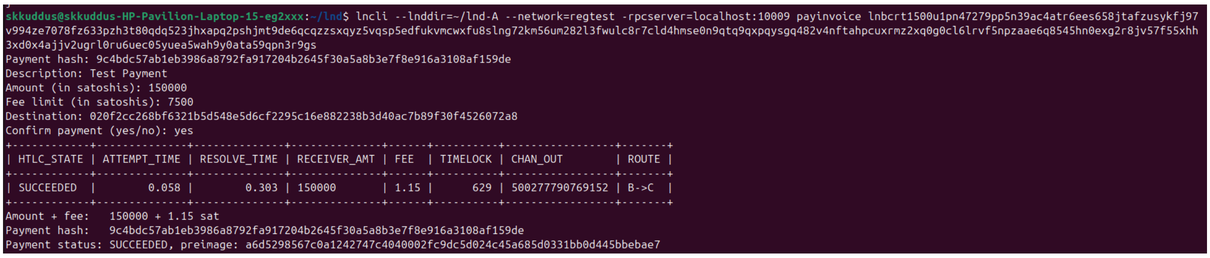
```
lncli --lnddir=~/lnd-A --network=regtest -rpcserver=localhost
    :10009 payinvoice <invoice_string>
```

replace this invoice_string by

```
    lnbcrt1500u1pn47279pp5n39ac4atr6ees658jtafzusykfj97v994ze7078
    fz633pzh3t80qdq523jhxapq2pshjmt9de6qcqzzsxqyz5vqsp5edfukvmcwx
    fu8slng72km56um282l3fwulc8r7cld4hmse0n9qtq9qxpqysgq482v4nftahp

    cuxrmz2xq0g0cl6lrvf5npzaae6q8545hn0exg2r8jv57f55xhh3xd0x4ajjv2

    ugrl0ru6uec05yuea5wah9y0ata59qpn3r9gs
```



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest -rpcserver=localhost:10009 payinvoice lnbcrt1500u1pn47279pp5n39ac4atr6ees658jtafzusykfj97
v994ze7078fz633pzh3t80qdq523jhxapq2pshjmt9de6qcqzzsxqyz5vqsp5edfukvmcwxfu8slng72km56um282l3fwulc8r7cld4hmse0n9qtq9qxpqysgq482v4nftahpcuxrmz2xq0g0cl6lrvf5npzaae6q8545hn0exg2r8jv57f55xhh
3xd0x4ajjv2ugrl0ru6uec05yuea5wah9y0ata59qpn3r9gs
Payment hash: 9c4bdc57ab1eb3986a8792fa917204b2645f30a5a8b3e7f8e916a3108af159de
Description: Test Payment
Amount (in satoshis): 150000
Fee limit (in satoshis): 7500
Destination: 020f2cc268bf6321b5d548e5d6cf2295c16e882238b3d40ac7b89f30f4526072a8
Confirm payment (yes/no): yes
+------------+--------------+--------------+--------------+------+----------+-----------------+-------+
| HTLC_STATE | ATTEMPT_TIME | RESOLVE_TIME | RECEIVER_AMT | FEE  | TIMELOCK | CHAN_OUT        | ROUTE |
+------------+--------------+--------------+--------------+------+----------+-----------------+-------+
| SUCCEEDED  |        0.058 |        0.303 | 150000       | 1.15 |      629 | 500277790769152 | B->C  |
+------------+--------------+--------------+--------------+------+----------+-----------------+-------+
Amount + fee:   150000 + 1.15 sat
Payment hash:   9c4bdc57ab1eb3986a8792fa917204b2645f30a5a8b3e7f8e916a3108af159de
Payment status: SUCCEEDED, preimage: a6d5298567c0a1242747c4040002fc9dc5d024c45a685d0331bb0d445bbebae7
```

Observation: Payment Routing in LDN

When a payment is initiated, the system generates the following payment hash:

```
9c4bdc57ab1eb3986a8792fa917204b2645f30a5a8b3e7f8e916a3108af159de
```

10.3 How to Monitor Payment Status

Replace `<payment_request>` with the actual payment request string. This command will begin the payment process.

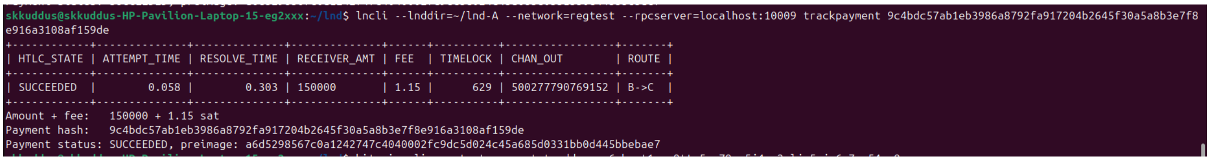
10.3.1 Monitor Payment in Real Time

Use the `trackpayment` command to monitor the status of a payment:

```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
   :10009 trackpayment <payment_hash>
```

Listing 54: Monitor Payment Status

Replace `<payment_hash>` with the unique hash of the payment. This will track the payment in real time and provide detailed information about its progress.



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 trackpayment 9c4bdc57ab1eb3986a8792fa917204b2645f30a5a8b3e7f8
e916a3108af159de
+------------+--------------+--------------+--------------+------+----------+-----------------+-------+
| HTLC_STATE | ATTEMPT_TIME | RESOLVE_TIME | RECEIVER_AMT | FEE  | TIMELOCK | CHAN_OUT        | ROUTE |
+------------+--------------+--------------+--------------+------+----------+-----------------+-------+
| SUCCEEDED  |        0.058 |        0.303 | 150000       | 1.15 |      629 | 500277790769152 | B->C  |
+------------+--------------+--------------+--------------+------+----------+-----------------+-------+
Amount + fee:   150000 + 1.15 sat
Payment hash:   9c4bdc57ab1eb3986a8792fa917204b2645f30a5a8b3e7f8e916a3108af159de
Payment status: SUCCEEDED, preimage: a6d5298567c0a1242747c4040002fc9dc5d024c45a685d0331bb0d445bbebae7
```

11 AMP (Atomic Multipath Payments)?

Atomic Multipath Payments (AMP) is a feature in the Lightning Network that allows large payments to be split into smaller parts (called *shards*) and sent through multiple routes to the recipient. The shards are combined at the recipient's end to form the full payment.

AMP enhances the usability of the Lightning Network by improving:

- **Liquidity Utilization**: Large payments leverage liquidity across multiple channels.
- **Privacy**: Payment shards are routed through different paths, increasing anonymity.
- **Reliability**: Reduces the risk of failed payments by distributing the load across multiple routes.

—

11.1 How AMP Works

1. The sender creates an invoice specifying the total amount to be paid.
2. The payment is divided into multiple shards.
3. Each shard follows a different route to the recipient.
4. The recipient combines these shards using preimages derived from a shared secret (hash).

11.2 Steps to Use AMP

11.2.1 Enable AMP Support

Ensure that your `lnd` version is up to date:

```
lncli --version
```

Listing 55: Check LND Version

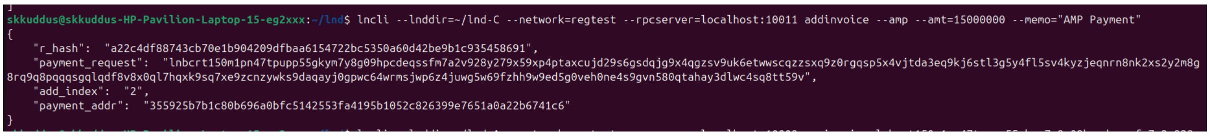
Use `lnd` version 0.13 or later for AMP support.

11.2.2 Generate an AMP Invoice

Create an AMP-compatible invoice using the `addinvoice` command with the `--amp` flag:

```
lncli --lnddir=~/lnd-C --network=regtest --rpcserver=localhost
   :10011 addinvoice --amp --amt=15000000 --memo="AMP Payment"
```

Listing 56: Create AMP Invoice



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-C --network=regtest --rpcserver=localhost:10011 addinvoice --amp --amt=15000000 --memo="AMP Payment"
{
    "r_hash":  "a22c4df88743cb70e1b904209dfbaa6154722bc5350a60d42be9b1c935458691",
    "payment_request":  "lnbcrt150m1pn47tpupp55gkym7y8g09hpcdeqssfm7a2v928y279x59xp4ptaxcujd29s6gsdqjg9x4qgzsv9uk6etwwscqzzsxq9z0rgqsp5x4vjtda3eq9kj6stl3g5y4fl5sv4kyzjeqnrn8nk2xs2y2m8g
8rq9q8pqqqsgqlqdf8v8x0ql7hqxk9sq7xe9zcnzywks9daqayj0gpwc64wrmsjwp6z4juwg5w69fzhh9w9ed5g0veh0ne4s9gvn580qtahay3dlwc4sq8tt59v",
    "add_index":  "2",
    "payment_addr":  "355925b7b1c80b696a0bfc5142553fa4195b1052c826399e7651a0a22b6741c6"
}
```

- `payment_request`: The AMP-enabled payment request to be shared with the payer.
- `payment_addr`: A unique payment address for this invoice.
- `r_hash`: The payment hash for the invoice.

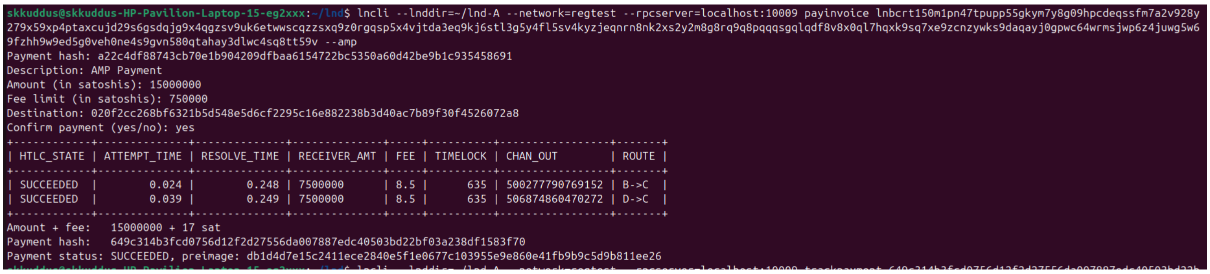
11.2.3 Pay an AMP Invoice

To pay an AMP invoice, use the following command from the sender's node:

```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
   :10009 payinvoice <payment_request> --amp
```

Listing 57: Pay AMP Invoice

Replace `<payment_request>` with the AMP-enabled payment request string.



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 payinvoice lnbcrt150m1pn47tpupp55gkym7y8g09hpcdeqssfm7a2v928y
279x59xp4ptaxcujd29s6gsdqjg9x4qgzsv9uk6etwwscqzzsxq9z0rgqsp5x4vjtda3eq9kj6stl3g5y4fl5sv4kyzjeqnrn8nk2xs2y2m8g8rq9q8pqqqsgqlqdf8v8x0ql7hqxk9sq7xe9zcnzywks9daqayj0gpwc64wrmsjwp6z4juwg5w6
9fzhh9w9ed5g0veh0ne4s9gvn580qtahay3dlwc4sq8tt59v --amp
Payment hash: a22c4df88743cb70e1b904209dfbaa6154722bc5350a60d42be9b1c935458691
Description: AMP Payment
Amount (in satoshis): 15000000
Fee limit (in satoshis): 750000
Destination: 020f2cc268bf6321b5d548e5d6cf2295c16e882238b3d40ac7b89f30f4526072a8
Confirm payment (yes/no): yes
+------------+--------------+--------------+--------------+-----+----------+-----------------+-------+
| HTLC_STATE | ATTEMPT_TIME | RESOLVE_TIME | RECEIVER_AMT | FEE | TIMELOCK | CHAN_OUT        | ROUTE |
+------------+--------------+--------------+--------------+-----+----------+-----------------+-------+
| SUCCEEDED  |        0.024 |        0.248 | 7500000      | 8.5 |      635 | 500277790769152 | B->C  |
| SUCCEEDED  |        0.039 |        0.249 | 7500000      | 8.5 |      635 | 506874860470272 | D->C  |
+------------+--------------+--------------+--------------+-----+----------+-----------------+-------+
Amount + fee:   15000000 + 17 sat
Payment hash:   649c314b3fcd0756d12f2d27556da007887edc40503bd22bf03a238df1583f70
Payment status: SUCCEEDED, preimage: db1d4d7e15c2411ece2840e5f1e0677c103955e9e860e41fb9b9c5d9b811ee26
```

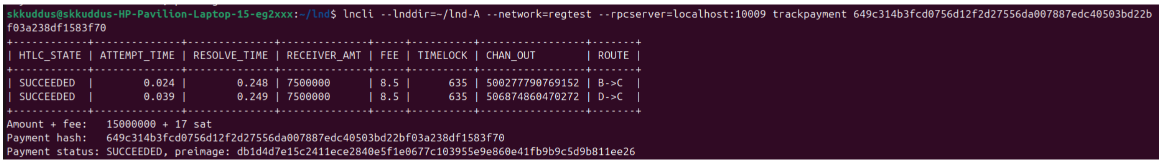
The Lightning Network Daemon (LDN) will automatically find a route (multi-hop) from Node A to Node C through Node B and/or Node D.

As the channel contains only `15000000 satoshis`, the LDN will split the payment into two parts:

- `750000 satoshis` through the route `A` → `B` → `C`.
- `750000 satoshis` through the route `A` → `D` → `C`.

This ensures the total payment is successfully delivered to the destination using the available channels and balances.

11.2.4 AMP status



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 trackpayment 649c314b3fcd0756d12f2d27556da007887edc40503bd22b
f03a238df1583f70
+------------+--------------+--------------+--------------+-----+----------+-----------------+-------+
| HTLC_STATE | ATTEMPT_TIME | RESOLVE_TIME | RECEIVER_AMT | FEE | TIMELOCK | CHAN_OUT        | ROUTE |
+------------+--------------+--------------+--------------+-----+----------+-----------------+-------+
| SUCCEEDED  |        0.024 |        0.248 | 7500000      | 8.5 |      635 | 500277790769152 | B->C  |
| SUCCEEDED  |        0.039 |        0.249 | 7500000      | 8.5 |      635 | 506874860470272 | D->C  |
+------------+--------------+--------------+--------------+-----+----------+-----------------+-------+
Amount + fee:   15000000 + 17 sat
Payment hash:   649c314b3fcd0756d12f2d27556da007887edc40503bd22bf03a238df1583f70
Payment status: SUCCEEDED, preimage: db1d4d7e15c2411ece2840e5f1e0677c103955e9e860e41fb9b9c5d9b811ee26
```

AMP Vs MPP

Atomic Multi-path Payments (AMP) differ from existing *Multi-path Payments (MPP)* in that they are atomic, meaning that despite being routed through separate paths, they are either settled in full or not settled at all.

In MPPs, all shards use the same payment hash, making the individual routes easily correlatable and prone to only partial settlement. By contrast, AMPs avoid these issues, ensuring complete settlement or none at all.

Using AMP, it is possible to make payments safely by only knowing the public key of the recipient. Additionally, AMP enables the creation of reusable invoices, which can facilitate traditional subscriptions. Such invoices can also be published without security implications, making them suitable for use cases such as static donation invoices.

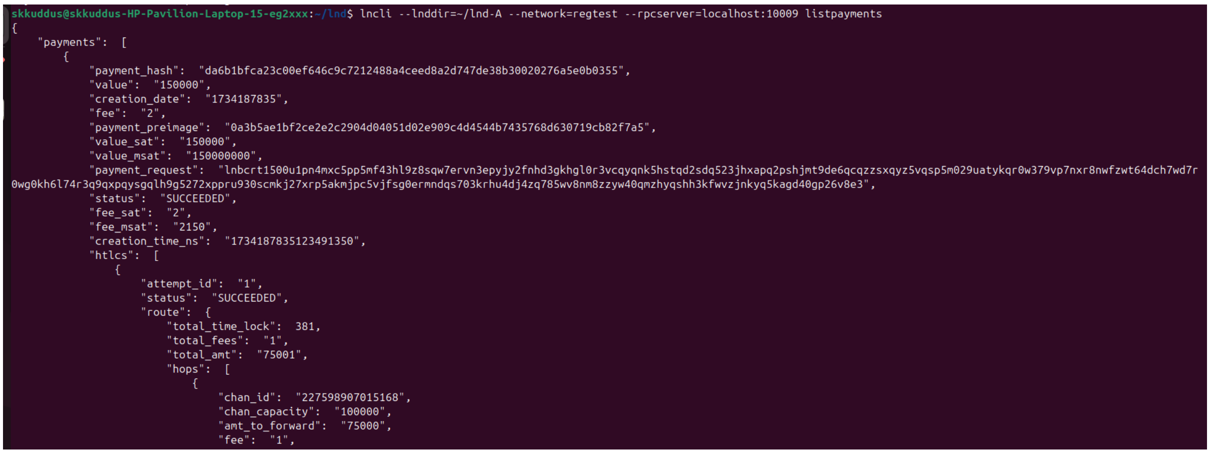
12 List Payments in the Lightning Network

To view all outgoing payments made by a node in the Lightning Network, use the `listpayments` command.

```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
    :10009 listpayments
```

Listing 58: List All Payments

Output



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost:10009 listpayments
{
    "payments":  [
        {
            "payment_hash":  "da6b1bfca23c00ef646c9c7212488a4ceed8a2d747de38b30020276a5e0b0355",
            "value":  "150000",
            "creation_date":  "1734187835",
            "fee":  "2",
            "payment_preimage":  "0a3b5ae1bf2ce2e2c2904d04051d02e909c4d4544b7435768d630719cb82f7a5",
            "value_sat":  "150000",
            "value_msat":  "150000000",
            "payment_request":  "lnbcrt1500u1pn4mxc5pp5mf43hl9z8sqw7ervn3epyjy2fnhd3gkhgl0r3vcqyqnk5hstqd2sdq523jhxapq2pshjmt9de6qcqzzsxqyz5vqsp5m029uatykqr0w379vp7nxr8nwfzwt64dch7wd7r
0wg0kh6l74r3q9qxpqysgqlh9g5272xppru930scmkj27xrp5akmjpc5vjfsg0ermndqs703krhu4dj4zq785wv8nm8zzyw40qmzhyqshh3kfwvzjnkyq5kagd40gp26v8e3",
            "status":  "SUCCEEDED",
            "fee_sat":  "2",
            "fee_msat":  "2150",
            "creation_time_ns":  "1734187835123491350",
            "htlcs":  [
                {
                    "attempt_id":  "1",
                    "status":  "SUCCEEDED",
                    "route":  {
                        "total_time_lock":  381,
                        "total_fees":  "1",
                        "total_amt":  "75001",
                        "hops":  [
                            {
                                "chan_id":  "227598907015168",
                                "chan_capacity":  "100000",
                                "amt_to_forward":  "75000",
                                "fee":  "1",
```

13 Closing Channels

To close channels for each node in the Lightning Network, use the following commands. Replace `<funding_txid>` with the transaction ID of the respective channel.

13.1 Close Channel for Nodes

Close Channel for Node A

```
lncli --lnddir=~/lnd-A --network=regtest --rpcserver=localhost
   :10009 closechannel --funding_txid=<funding_txid_A>
```

Listing 59: Close Channel for Node A

funding_txid_A:dee27f90cecfecb5b5f9bcaf75b5e615efa6135e38bd6be11ee60d697b6b1f3e

Close Channel for Node B

```
lncli --lnddir=~/lnd-B --network=regtest --rpcserver=localhost
   :10010 closechannel --funding_txid=<funding_txid_B>
```

Listing 60: Force Close Channel for Node B

funding_txid_B: 4863b5a2c42e12d37f9231410ce31a010483dde92c66100133565a0f590a7d66

Close Channel for Node C

```
lncli --lnddir=~/lnd-C --network=regtest --rpcserver=localhost
   :10011 closechannel --funding_txid=<funding_txid_C>
```

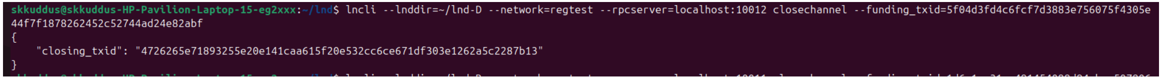
Listing 61: Force Close Channel for Node C

Close Channel for Node D

```
lncli --lnddir=~/lnd-D --network=regtest --rpcserver=localhost
   :10012 closechannel --funding_txid=<funding_txid_D>
```

Listing 62: Force Close Channel for Node D

funding_txid_D: 5f04d3fd4c6fcf7d3883e756075f4305e44f7f1878262452c52744ad24e82abf

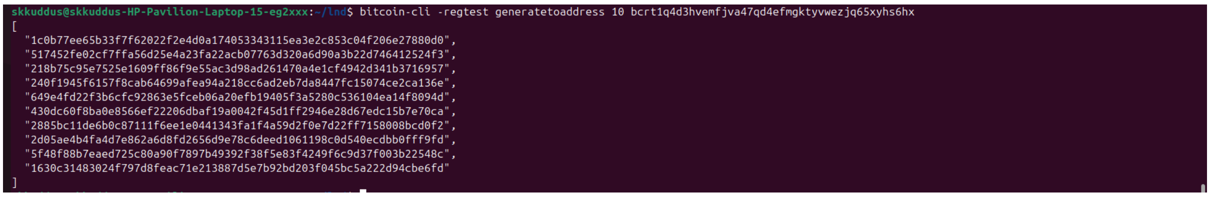


```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ lncli --lnddir=~/lnd-D --network=regtest --rpcserver=localhost:10012 closechannel --funding_txid=5f04d3fd4c6fcf7d3883e756075f4305e
44f7f1878262452c52744ad24e82abf
{
    "closing_txid": "4726265e71893255e20e141caa615f20e532cc6ce671df303e1262a5c2287b13"
}
```

13.2 Finalize Closure

Mine additional blocks and verify that the funds have been returned to our bitcoin regtest wallet and reflect the txn that we made off chain.

```
bitcoin-cli -regtest generatetoaddress 10
   bcrt1q4d3hvemfjva47qd4efmgktyvwezjq65xyhs6hx
```



```
skkuddus@skkuddus-HP-Pavilion-Laptop-15-eg2xxx:~/lnd$ bitcoin-cli -regtest generatetoaddress 10 bcrt1q4d3hvemfjva47qd4efmgktyvwezjq65xyhs6hx
[
  "1c0b77ee65b33f7f62022f2e4d0a174053343115ea3e2c853c04f206e27880d0",
  "517452fe02cf7ffa56d25e4a23fa22acb07763d320a6d90a3b22d746412524f3",
  "218b75c95e7525e1609ff86f9e55ac3d98ad261470a4e1cf4942d341b3716957",
  "240f1945f6157f8cab64699afea94a218cc6ad2eb7da8447fc15074ce2ca136e",
  "649e4fd22f3b6cfc92863e5fceb06a20efb19405f3a5280c536104ea14f8094d",
  "430dc60f8ba0e8566ef22206dbaf19a0042f45d1ff2946e28d67edc15b7e70ca",
  "2885bc11de6b0c87111f6ee1e0441343fa1f4a59d2f0e7d22ff7158008bcd0f2",
  "2d05ae4b4fa4d7e862a6d8fd2656d9e78c6deed1061198c0d540ecdbb0fff9fd",
  "5f48f88b7eaed725c80a90f7897b49392f38f5e83f4249f6c9d37f003b22548c",
  "1630c31483024f797d8feac71e213887d5e7b92bd203f045bc5a222d94cbe6fd"
]
```

14 Conclusion

Conclusion

In this project, we successfully implemented and explored key functionalities of the Lightning Network Daemon (LND), focusing on creating and managing Lightning channels, routing multi-hop payments, and comparing payment methods such as Multi-path Payments (MPP) and Atomic Multi-path Payments (AMP). By utilizing a `regtest` environment, we ensured a secure and controlled setup for experimenting with various payment configurations.

We demonstrated the creation of multiple channels between nodes and effectively established multi-hop routes to facilitate payments. The implementation of MPP and AMP highlighted the evolution of payment mechanisms in the Lightning Network, with AMP proving to be a secure and atomic alternative, ensuring complete settlement without correlation risks.

References

- Blockchain Commons, ”Learning Bitcoin from the Command Line,” GitHub Repository. Available at: https://github.com/BlockchainCommons/Learning-Bitcoin-from-the-C tree/master