Peer to Peer

Layers

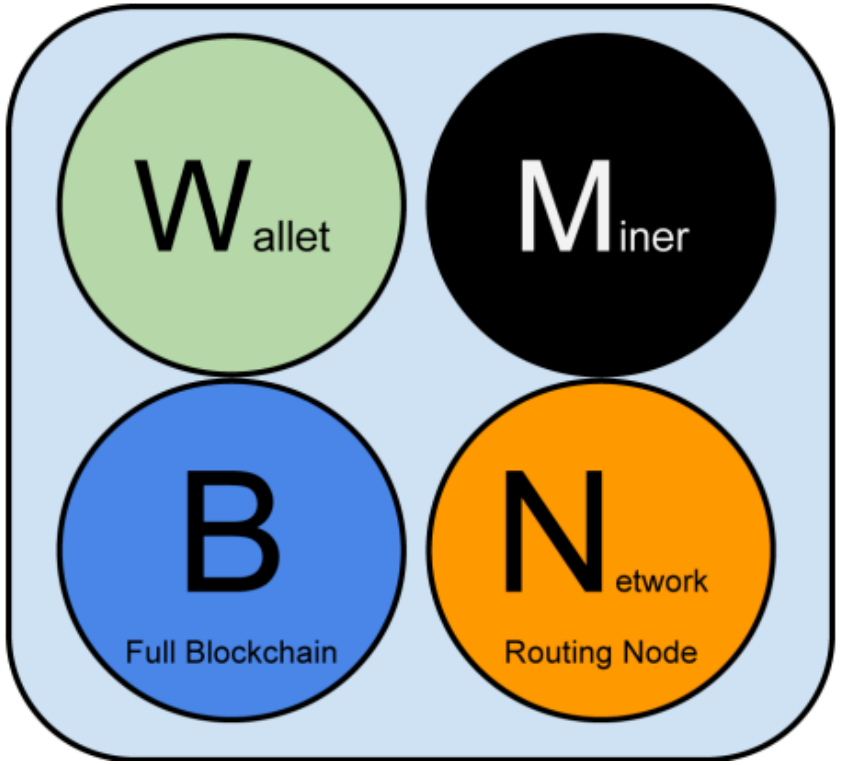
Token xShop, EPSCoin

Protocol Bitcoin, Etherum, NEO

Blockchain

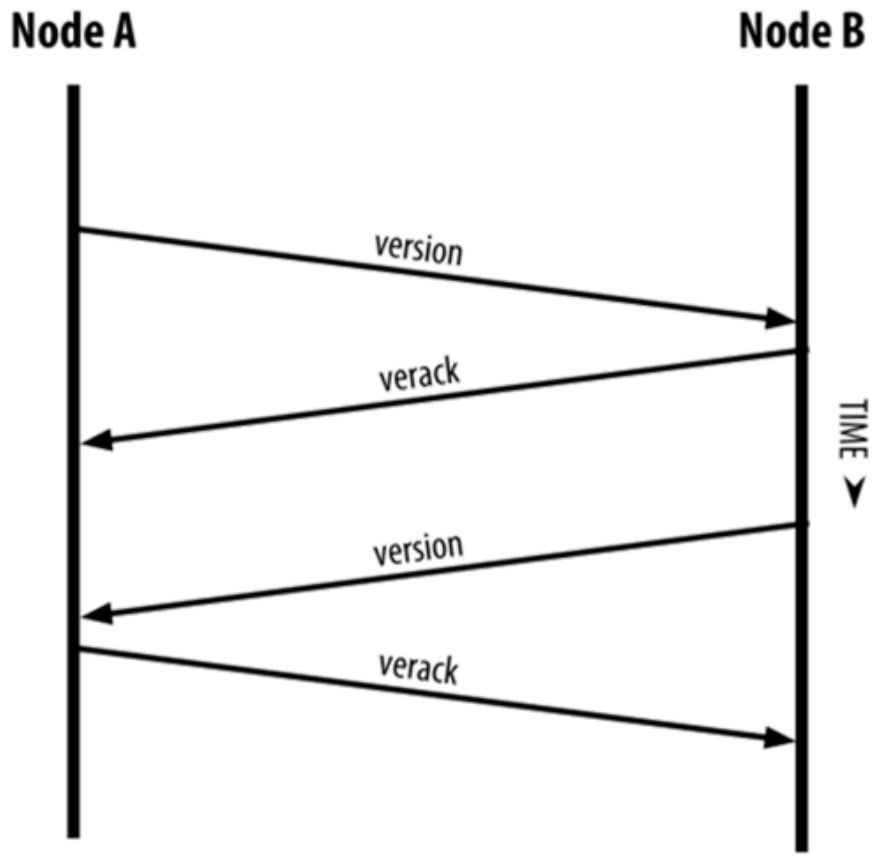
Node Types and Roles

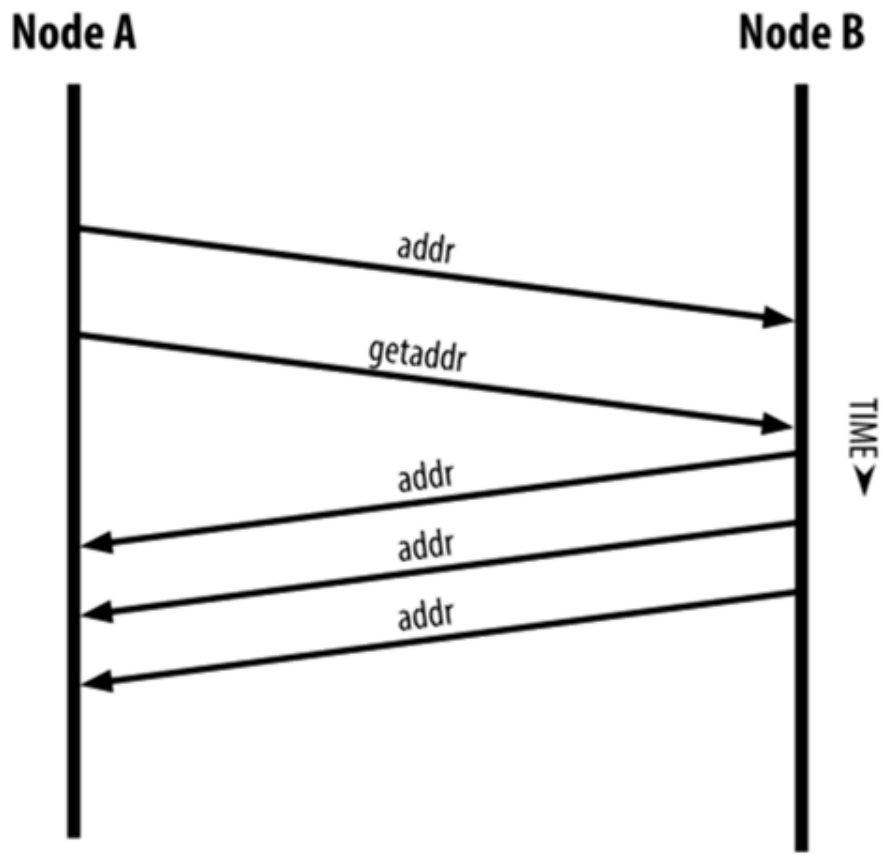
- depending on the functionality
  - routing,
  - the blockchain database,
  - mining, and
  - wallet services.

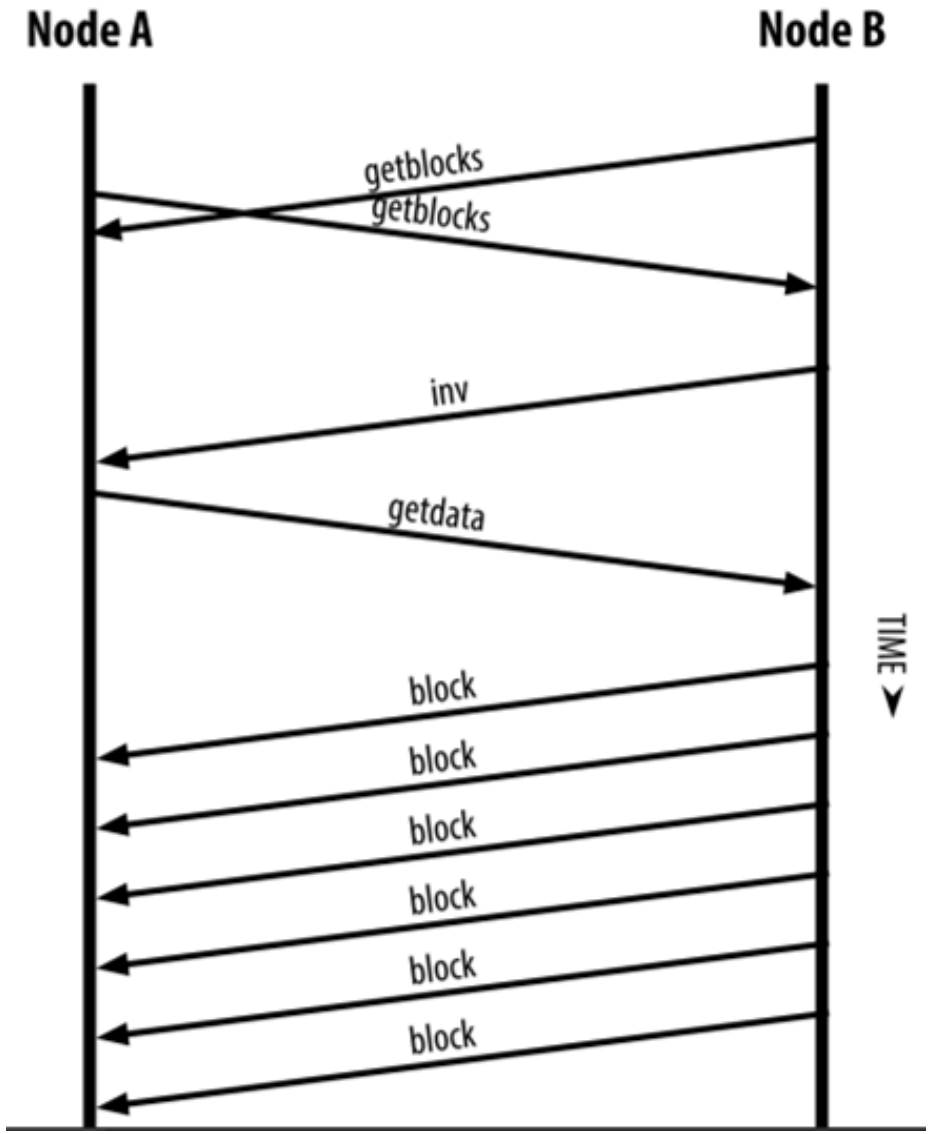


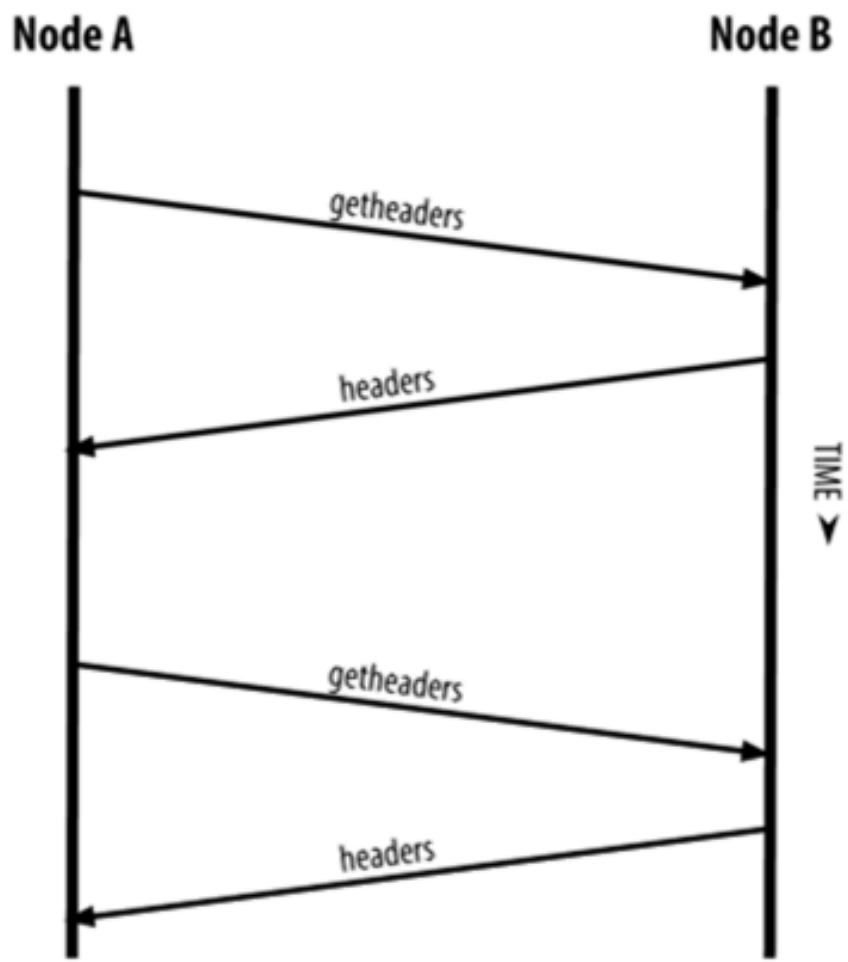
How does a new node find peers?

- DNS seeds (static list of IP addresses)
- Once one or more connections are established, the new node will send an address message containing its own IP address to its neighbors
- The neighbors will, in turn, forward the address message to their neighbors


Node A
Node B
version
verack
TIME
version
verack


Node A
Node B
addr
getaddr
addr
addr
addr
TIME


Node A
Node B
getblocks
getblocks
inv
getdata
block
block
block
block
block
block
TIME


Node A
Node B
getheaders
headers
TIME
getheaders
headers

Code (Python)

```
import socket
import threading

HEADER = 64
PORT = 5050
NODE = socket.gethostbyname(socket.gethostname())
ADDR = (NODE, PORT)

FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!exit"
```

Code (Python)

```
node = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    node.bind(ADDR)
    print("[STARTING] Node starting...")
    start()
```

Code (Python)

```
def start():
    node.listen()
    print(f"[LISTENING] Node is listening on {NODE}")
    while True:
        conn, addr = node.accept()
        thread = threading.Thread(target=handle_myPeer, args=(conn, addr))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {threading.activeCount() -1}")
```

Code (Python)

```
def handle_myPeer(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")
    connected = True
    while connected:
        msg_length = conn.recv(HEADER).decode(FORMAT)
        if msg_length:
            msg_length = int(msg_length)
            msg = conn.recv(msg_length).decode(FORMAT)
            if msg == DISCONNECT_MESSAGE:
                connected = False
            print(f"[{addr}] {msg}")
```

Code (Python)

```
def send(msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' ' * (HEADER - len(send_length))
    myPeer.send(send_length)
    myPeer.send(message)
    print(myPeer.recv(2048).decode(FORMAT))
```

Code (Python)

```
try:
    myPeer = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    myPeer.connect(ADDR)
    send("version")
    peerconnected = True
    while peerconnected:
        msg = input("Msg>>")
        send(msg)
        if msg == DISCONNECT_MESSAGE:
            peerconnected = False
```