Assignment 2 report

Introduction

This assignment involves building a P2P file transfer system by implementing a simplified Reliable Data Transfer (RDT) protocol and a Distance Vector (DV) routing algorithm at the application layer. Using UDP, the system ensures reliable file transmission and supports multi-hop routing between peers.

This code implements a peer-to-peer file transfer system over UDP with two main features:

- **Reliable Data Transfer (RDT):** Files are split into segments with sequence numbers, sent using a sliding window. Lost or corrupted packets are detected and retransmitted based on ACKs and timeouts.
- **Distance Vector (DV) Routing:** Peers exchange routing information to compute shortest paths. Messages are forwarded across multiple hops based on updated link costs.

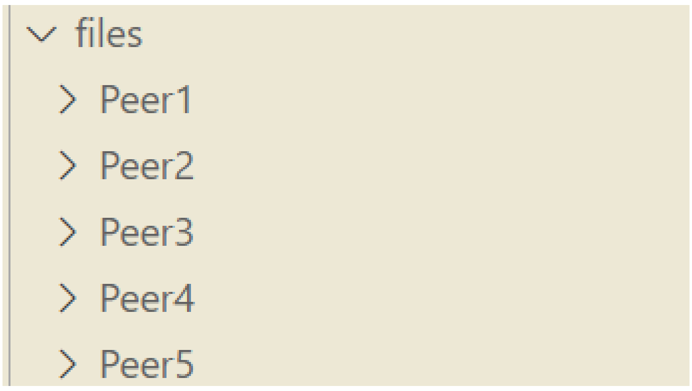
The system simulates unreliable networks with packet loss and errors, and supports user commands to send files, check routes, and monitor file reception.

Testing

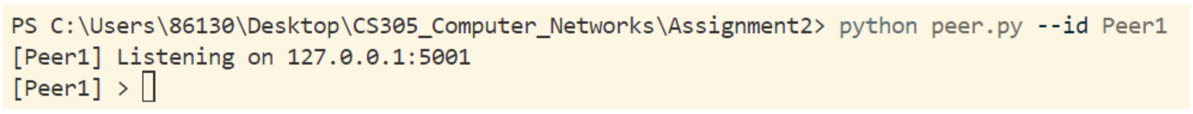
1. running five peers.

```
#take peer1 as an example
python peer.py --id Peer1
```

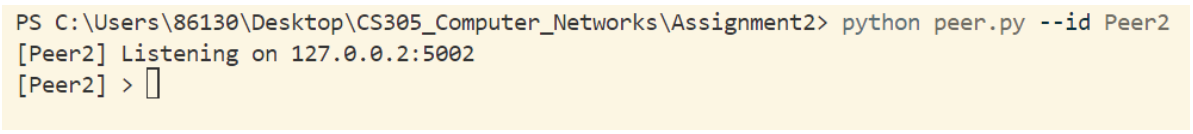
After entering a command, a `files/Peer_{id}` directory is created.



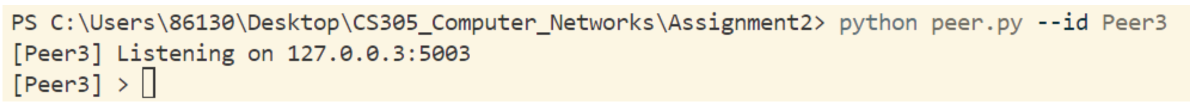
And we could enter the command mode of each peer.



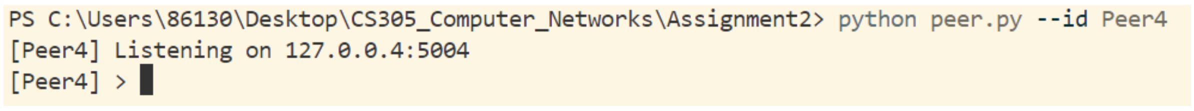
```
PS C:\Users\86130\Desktop\CS305_Computer_Networks\Assignment2> python peer.py --id Peer1
[Peer1] Listening on 127.0.0.1:5001
[Peer1] > 
```



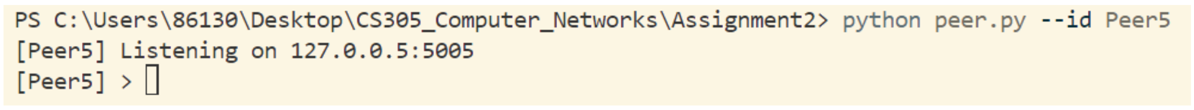
```
PS C:\Users\86130\Desktop\CS305_Computer_Networks\Assignment2> python peer.py --id Peer2
[Peer2] Listening on 127.0.0.2:5002
[Peer2] > 
```



```
PS C:\Users\86130\Desktop\CS305_Computer_Networks\Assignment2> python peer.py --id Peer3
[Peer3] Listening on 127.0.0.3:5003
[Peer3] > 
```



```
PS C:\Users\86130\Desktop\CS305_Computer_Networks\Assignment2> python peer.py --id Peer4
[Peer4] Listening on 127.0.0.4:5004
[Peer4] >
```

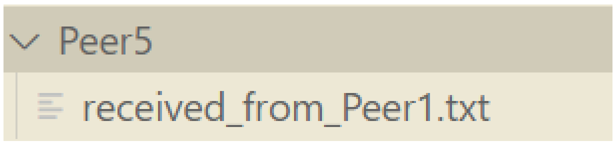


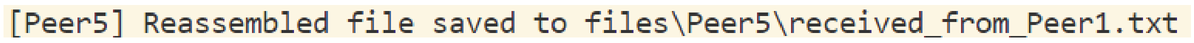
```
PS C:\Users\86130\Desktop\CS305_Computer_Networks\Assignment2> python peer.py --id Peer5
[Peer5] Listening on 127.0.0.5:5005
[Peer5] >
```

2. send input.txt to Peer5 in Peer1 terminal

```
send Peer5 input.txt
```

After the command, we could see a file named `received_from_Peer1.txt` is created under `files/Peer5` , which is exactly the same as the file `input.txt`.

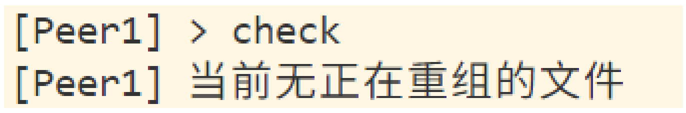




```
[Peer5] Reassembled file saved to files\Peer5\received_from_Peer1.txt
```

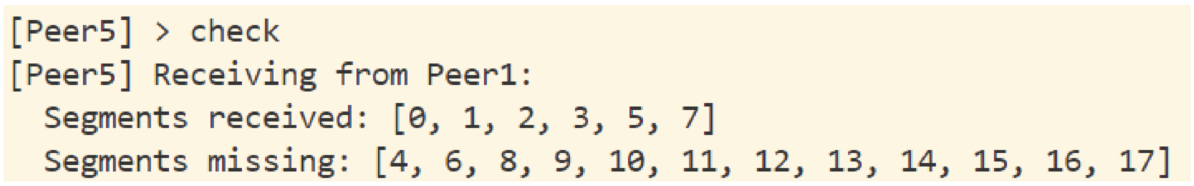
3. check command

If we input the command when no files are transmitting, we could see the following result.



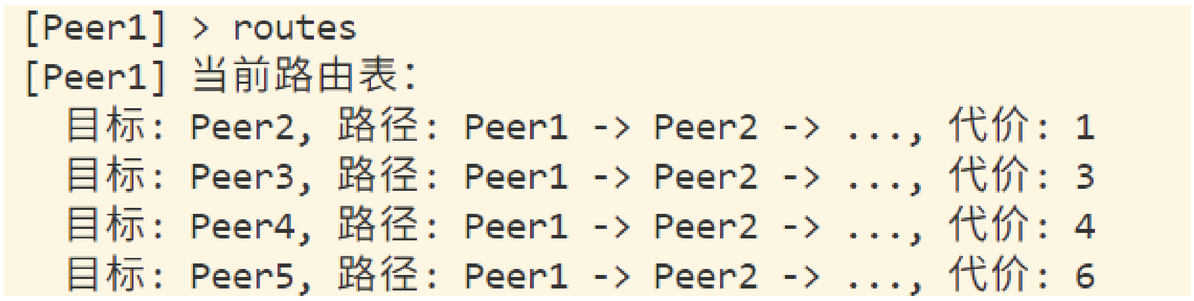
```
[Peer1] > check
[Peer1] 当前无正在重组的文件
```

if we input the command when some files are transmitting, we could see results in the following

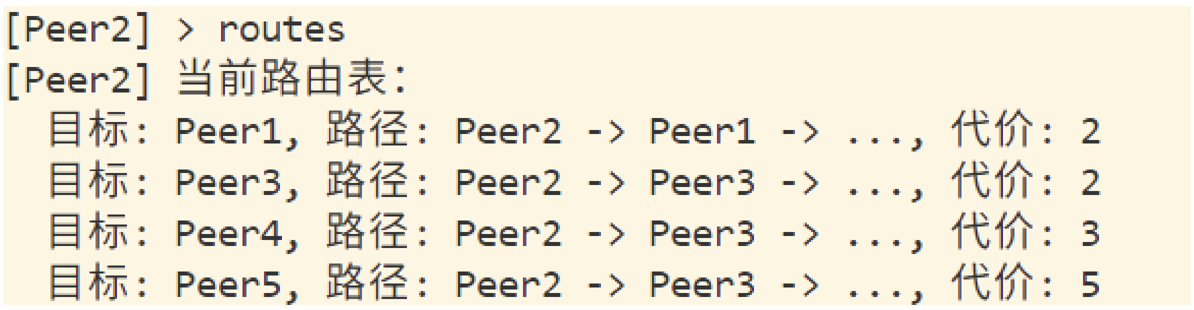


```
[Peer5] > check
[Peer5] Receiving from Peer1:
  Segments received: [0, 1, 2, 3, 5, 7]
  Segments missing: [4, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
```

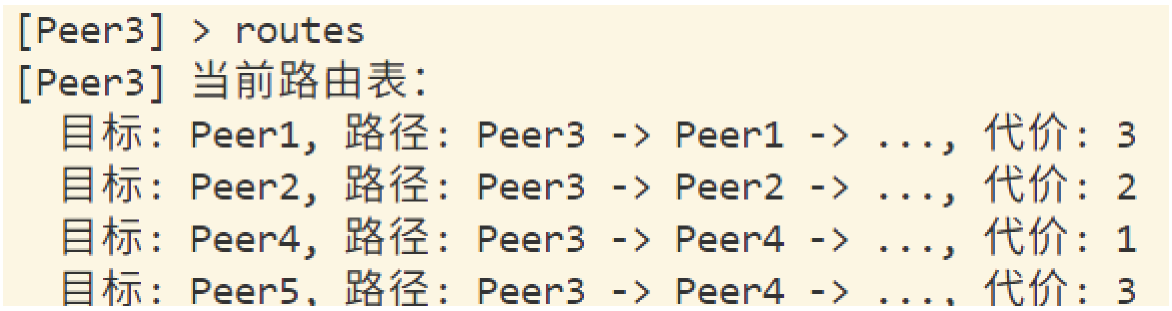
4. routes command



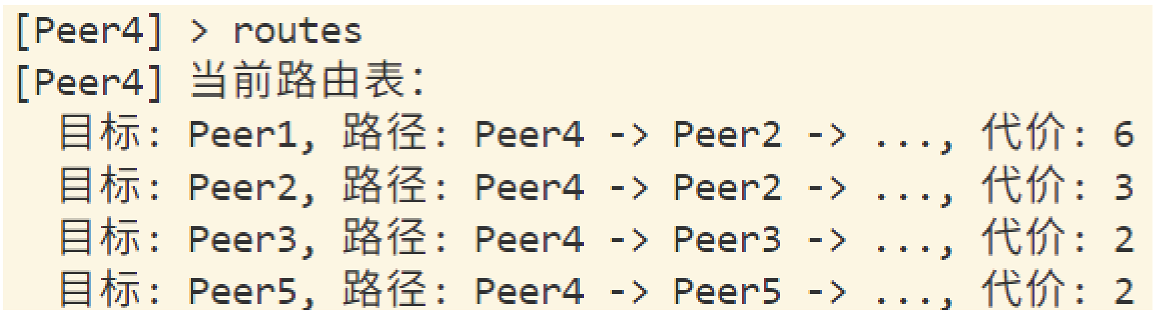
```
[Peer1] > routes
[Peer1] 当前路由表:
  目标: Peer2, 路径: Peer1 -> Peer2 -> ..., 代价: 1
  目标: Peer3, 路径: Peer1 -> Peer2 -> ..., 代价: 3
  目标: Peer4, 路径: Peer1 -> Peer2 -> ..., 代价: 4
  目标: Peer5, 路径: Peer1 -> Peer2 -> ..., 代价: 6
```



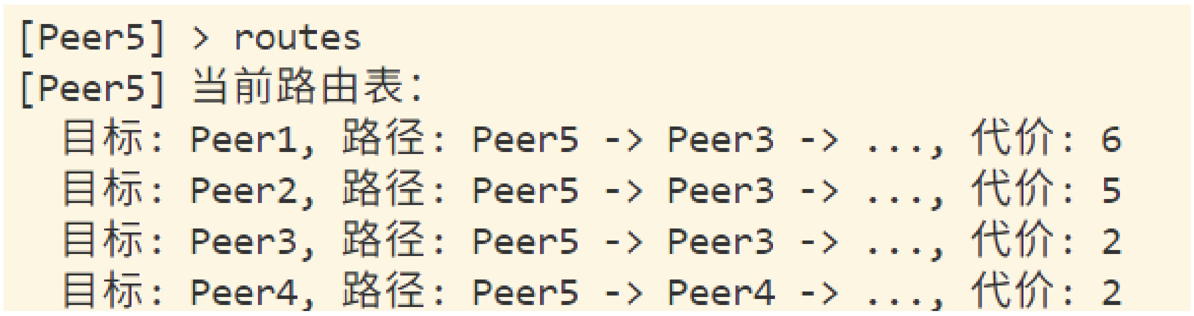
```
[Peer2] > routes
[Peer2] 当前路由表:
  目标: Peer1, 路径: Peer2 -> Peer1 -> ..., 代价: 2
  目标: Peer3, 路径: Peer2 -> Peer3 -> ..., 代价: 2
  目标: Peer4, 路径: Peer2 -> Peer3 -> ..., 代价: 3
  目标: Peer5, 路径: Peer2 -> Peer3 -> ..., 代价: 5
```



```
[Peer3] > routes
[Peer3] 当前路由表:
  目标: Peer1, 路径: Peer3 -> Peer1 -> ..., 代价: 3
  目标: Peer2, 路径: Peer3 -> Peer2 -> ..., 代价: 2
  目标: Peer4, 路径: Peer3 -> Peer4 -> ..., 代价: 1
  目标: Peer5, 路径: Peer3 -> Peer4 -> ..., 代价: 3
```

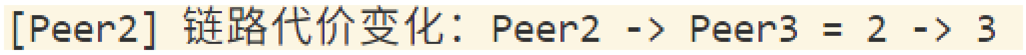


```
[Peer4] > routes
[Peer4] 当前路由表:
  目标: Peer1, 路径: Peer4 -> Peer2 -> ..., 代价: 6
  目标: Peer2, 路径: Peer4 -> Peer2 -> ..., 代价: 3
  目标: Peer3, 路径: Peer4 -> Peer3 -> ..., 代价: 2
  目标: Peer5, 路径: Peer4 -> Peer5 -> ..., 代价: 2
```



```
[Peer5] > routes
[Peer5] 当前路由表:
  目标: Peer1, 路径: Peer5 -> Peer3 -> ..., 代价: 6
  目标: Peer2, 路径: Peer5 -> Peer3 -> ..., 代价: 5
  目标: Peer3, 路径: Peer5 -> Peer3 -> ..., 代价: 2
  目标: Peer4, 路径: Peer5 -> Peer4 -> ..., 代价: 2
```

link costs can randomly change like below:



```
[Peer2] 链路代价变化: Peer2 -> Peer3 = 2 -> 3
```

Conclusion

The project demonstrates a fully working reliable UDP file transfer protocol with Distance Vector routing in a simulated peer-to-peer network.

The code can successfully send and receive files and distance vector routing tables were correctly populated.