Executive Summary

- Goal: Enable peer-to-peer backup of files with regular heartbeats to a coordination server.

System Overview

- Roles
  - `OWNER`: Sends backup requests, splits files into chunks, and sends over TCP.
  - `STORAGE`: Listens for TCP chunk transfers, validates CRC, persists chunks, and sends UDP acks.
  - `Server`: Coordinates registration, heartbeats, and backup plans (optional in offline mode).
- Channels
  - UDP control: `REGISTER`, `HEARTBEAT`, `BACKUP_REQ`, `BACKUP_PLAN`, `CHUNK_OK`, `CHUNK_ERROR`, `BACKUP_DONE`.
  - TCP data: `SEND_CHUNK <header>` followed by raw bytes.
- Offline fallback: If the server does not provide a plan, owners use `backup.staticPeers` and `backup.chunkSize` from config.

Components

- `PeerMain`: CLI entry; wires services for heartbeat or backup.
- `PeerConfig`: Loads properties from a file (default `config.properties`; override via `-Dconfig` or `-Config` in the runner).
- `MessageCodec`: Encodes/decodes protocol messages.
- `UdpControlChannel`: Sends control messages; awaits acks via internal queues.
- `TcpChunkSender`: Opens TCP connection and sends `SEND_CHUNK` header + bytes.
- `BackupService`: Orchestrates requests, chunking, sending, ack waiting (5s), retries (up to 3), and final `BACKUP_DONE`.
- `StorageReceiverService`: Listens on TCP; validates checksums; writes chunks under `storage/<FileName>/chunk-<id>.bin`; sends `CHUNK_OK`.
- `HeartbeatService`: Periodic `HEARTBEAT` per `heartbeat.intervalSeconds`.
- `Crc32Util`: Computes CRC32 for file and chunks.

Protocol Reference

- UDP control
  - `REGISTER RQ# Name Role IP UDP_Port TCP_Port StorageMB`
  - `REGISTERED Name OK`
  - `HEARTBEAT Name ChunkCount Timestamp`
  - `BACKUP_REQ Name FileName FileSize CRC32`
  - `BACKUP_PLAN Name FileName ChunkSize PeerList`
  - `BACKUP_DENIED Name Reason`
  - `CHUNK_OK OwnerName FileName ChunkId`
  - `CHUNK_ERROR OwnerName FileName ChunkId Reason`
  - `BACKUP_DONE Name FileName TotalChunks`
- TCP data
  - `SEND_CHUNK OwnerName OwnerUdpPort FileName FileSize FullCRC ChunkId ChunkOffset ChunkSize ChunkCRC`
  - Then: exactly `ChunkSize` bytes.
- Ack behavior
  - Owner waits up to 5 seconds for `CHUNK_OK`/`CHUNK_ERROR`; retries up to 3 times; cancels (`CHUNK_CANCEL`) after failures; sends `BACKUP_DONE` when all chunks succeed.

Configuration

- How config is loaded
  - Default: `config.properties`; override with `-Dconfig=<file>` or runner `-Config <file>`.
- Keys (commonly used)
  - `peer.name` (e.g., `Alice`, `Bob`)
  - `peer.role` (`OWNER` or `STORAGE`)
  - `peer.ip`, `peer.udpPort`, `peer.tcpPort`, `peer.storageMB`
  - `server.host`, `server.udpPort`

  - `heartbeat.intervalSeconds`

  - `backup.staticPeers` format: `[Name@Host:TcpPort:UdpPort,...]`

  - `backup.chunkSize` (bytes) for offline chunking

- Example: Alice (Owner)

  - `backup.staticPeers=[Bob@127.0.0.1:6002:5002]`

  - `backup.chunkSize=65536`

  - `peer.udpPort=5001`, `peer.tcpPort=6001`

- Example: Bob (Storage)

  - `peer.udpPort=5002`, `peer.tcpPort=6002`, `peer.storageMB=512`

Demo

- Setup terminals

  - Terminal A → Bob (storage)

  - Terminal B → Alice (owner)

- Step 1: Build

  - `./build.ps1`

  - See: “Build succeeded. Classes at: out”

- Step 2: Start Bob (storage)

  - `./run_peer.ps1 heartbeat -Config config.bob.properties`

  - See: `REGISTER ... Bob STORAGE ...` and `STORAGE TCP receiver listening on port 6002`

  - Quick port check: `netstat -ano | findstr LISTENING | findstr :6002` → shows `LISTENING`

- Step 3: Prepare test file (if missing)

  - `@('Hello from P2PBRS','Backup validation run before production','Line 3: OK') | Set-Content -Path 'sample.txt' -Encoding ASCII`

  - Verify: `Get-Item 'sample.txt' | Format-Table Name,Length,FullName -AutoSize` → expect length 72

- Step 4: Run Alice (backup)

  - `./run_peer.ps1 backup sample.txt -Config config.alice.properties`
  - See:
    - `REGISTER ... Alice OWNER ...`
    - `BACKUP_REQ ... sample.txt ...`
    - `BACKUP_PLAN using local static peers, chunkSize=65536`
    - `SEND_CHUNK ... file=sample.txt id=0 size=72 -> 127.0.0.1:6002 (attempt 1)`
    - `CHUNK_OK ... file=sample.txt id=0`
    - `BACKUP_DONE ... sample.txt`
- Step 5: Verify artifacts
  - On disk: `storage\sample.txt\chunk-0.bin` exists; size 72 for the example.
  - Hash check (single-chunk case):
    - `Get-FileHash sample.txt -Algorithm MD5`
    - `Get-FileHash storage\sample.txt\chunk-0.bin -Algorithm MD5`
    - Hashes should match.

Troubleshooting (Common Issues)

- Build output missing
  - Symptom: `Build output 'out' not found`
  - Fix: run `./build.ps1` and retry.
- Storage not listening / connection refused
  - Ensure Bob is running; check port via `netstat -ano | findstr LISTENING | findstr :6002`.
  - Free port conflicts or change `peer.tcpPort` in `config.bob.properties`.
- Test file missing
  - Symptom: `BACKUP failed: sample.txt (The system cannot find the file specified)`
  - Fix: recreate `sample.txt` or use a real file path.
- Test-NetConnection appears stuck

  - Use `netstat` for quick confirmation.

- No `CHUNK_OK`

  - Verify storage logs show `SEND_CHUNK` and CRC OK; confirm UDP reachability to owner's `peer.udpPort`.

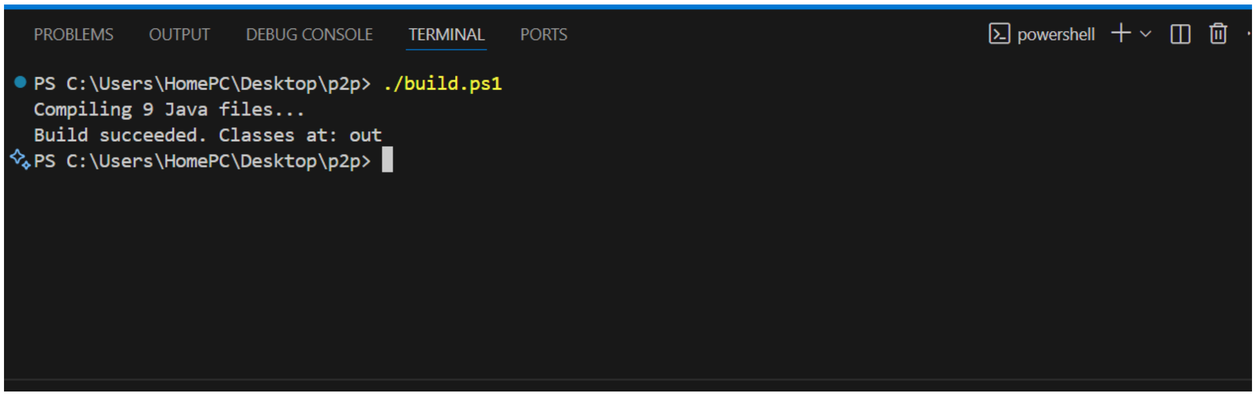
- CRC failures

  - Validate header fields and chunk size alignment; rebuild and retry.

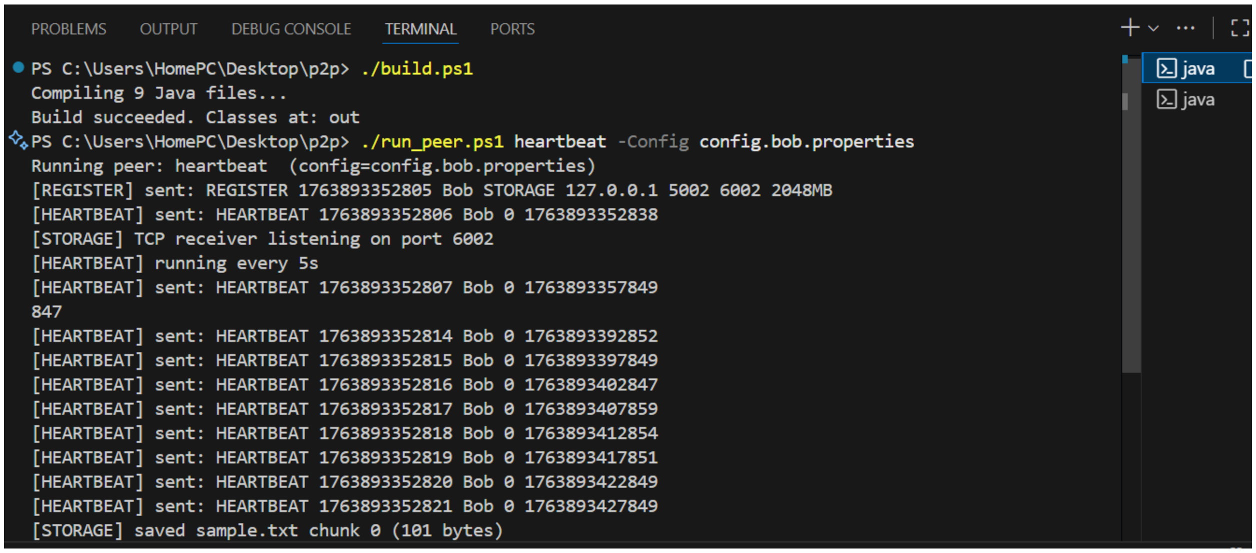
Notes

- `out/` is build output; delete and rebuild any time.

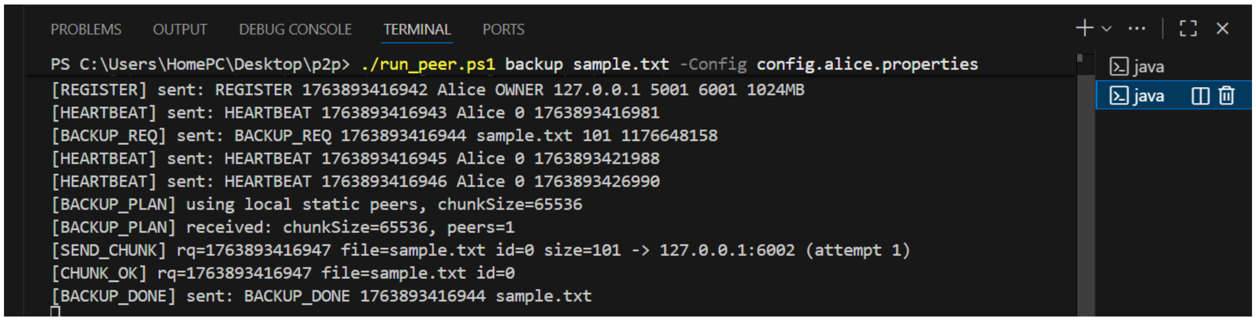
- `storage/` holds runtime chunk data; safe to delete per file for cleanup tests.



```
PS C:\Users\HomePC\Desktop\p2p> ./build.ps1
Compiling 9 Java files...
Build succeeded. Classes at: out
PS C:\Users\HomePC\Desktop\p2p>
```



```
PS C:\Users\HomePC\Desktop\p2p> ./build.ps1
Compiling 9 Java files...
Build succeeded. Classes at: out
PS C:\Users\HomePC\Desktop\p2p> ./run_peer.ps1 heartbeat -Config config.bob.properties
Running peer: heartbeat  (config=config.bob.properties)
[REGISTER] sent: REGISTER 1763893352805 Bob STORAGE 127.0.0.1 5002 6002 2048MB
[HEARTBEAT] sent: HEARTBEAT 1763893352806 Bob 0 1763893352838
[STORAGE] TCP receiver listening on port 6002
[HEARTBEAT] running every 5s
[HEARTBEAT] sent: HEARTBEAT 1763893352807 Bob 0 1763893357849
847
[HEARTBEAT] sent: HEARTBEAT 1763893352814 Bob 0 1763893392852
[HEARTBEAT] sent: HEARTBEAT 1763893352815 Bob 0 1763893397849
[HEARTBEAT] sent: HEARTBEAT 1763893352816 Bob 0 1763893402847
[HEARTBEAT] sent: HEARTBEAT 1763893352817 Bob 0 1763893407859
[HEARTBEAT] sent: HEARTBEAT 1763893352818 Bob 0 1763893412854
[HEARTBEAT] sent: HEARTBEAT 1763893352819 Bob 0 1763893417851
[HEARTBEAT] sent: HEARTBEAT 1763893352820 Bob 0 1763893422849
[HEARTBEAT] sent: HEARTBEAT 1763893352821 Bob 0 1763893427849
[STORAGE] saved sample.txt chunk 0 (101 bytes)
```



```
PS C:\Users\HomePC\Desktop\p2p> ./run_peer.ps1 backup sample.txt -Config config.alice.properties
[REGISTER] sent: REGISTER 1763893416942 Alice OWNER 127.0.0.1 5001 6001 1024MB
[HEARTBEAT] sent: HEARTBEAT 1763893416943 Alice 0 1763893416981
[BACKUP_REQ] sent: BACKUP_REQ 1763893416944 sample.txt 101 1176648158
[HEARTBEAT] sent: HEARTBEAT 1763893416945 Alice 0 1763893421988
[HEARTBEAT] sent: HEARTBEAT 1763893416946 Alice 0 1763893426990
[BACKUP_PLAN] using local static peers, chunkSize=65536
[BACKUP_PLAN] received: chunkSize=65536, peers=1
[SEND_CHUNK] rq=1763893416947 file=sample.txt id=0 size=101 -> 127.0.0.1:6002 (attempt 1)
[CHUNK_OK] rq=1763893416947 file=sample.txt id=0
[BACKUP_DONE] sent: BACKUP_DONE 1763893416944 sample.txt
```