# PSI z78
# TORRENTS PROJECT

*Józef Żyta 334705*

```
[20:40:47] (peer:45) DOWNLOAD SOURCES SUMMARY (chunks/bytes by peer):
[20:40:47] (peer:45)   172.21.78.5:10002 -> chunks=13 bytes=49580
[20:40:47]docker exec -it t_peer2 /app/tpeer seed example.bin
[20:54:00] (peer:8) ANNOUNCE ok file=example.bin full_size=65964 have_si
ze=65964 chunk=4096
torrents> docker exec -it t_peer3 /app/tpeer get example.bin /data/examp
le.bin
cker=t_tracker:9000
[20:54:04] (peer:8) seed[0]=172.21.78.3:10002
[20:54:04] (peer:8) QUERY file=example.bin -> size=65964 chunk=4096 nchu
nks=17 npeers=1
[20:54:04] (peer:8) GET resume: already_have_bytes=65964 already_have_ch
unks=17
[20:54:04] (peer:8) DOWNLOAD SOURCES SUMMARY (chunks/bytes by peer):
[20:54:04] (peer:8) GET complete file=example.bin out=/data/example.bin
torrents> docker exec -it t_peer3 /app/tpeer get example.bin /data/examp
le.bin
[20:54:09] (peer:18) GET start file=example.bin out=/data/example.bin tr
acker=t_tracker:9000
[20:54:09] (peer:18) seed[0]=172.21.78.3:10002
[20:54:09] (peer:18) QUERY file=example.bin -> size=65964 chunk=4096 nch
unks=17 npeers=1
[20:54:09] (peer:18) fetch: chunk=0 from 172.21.78.3:10002
[20:54:09] (peer:18) fetch: chunk=2 from 172.21.78.3:10002
```

# Project idea

A torrent system with tracker to keep the list of files available on peers, peers that act as servers to provide files in chunks. Peers also provide an app with “GET” and “SEED” methods to download file, or to announce file to tracker.

# Plan

Tracker:

- Keeps a list of files
- For each file keeps a list of peers (ip + port)
- For each peer keeps a bitfield (which chunks this peer has)
- Handles:
- ANNOUNCE: peer sends file meta and bitfield, tracker saves it
- QUERY: client asks for a file, tracker returns meta + peer list

Peer:

1. Serve
    a. Listens on TCP port
    b. Answers GET requests
    c. Reads chunk from local file and sends back DATA + bytes
2. Seed
    a. Checks local file size
    b. Builds bitfield based on file parts that exist on disk
    c. Sends ANNOUNCE to tracker
3. Get
    a. Sends QUERY to tracker
    b. Gets list of peers and their bitfields
    c. Downloads missing chunks from many peers (threads)
    d. Writes chunks into output file at right offsets
    e. Can resume if output file already exists

Common library (proto):

- TCP connect and listen helpers
- Read line and write all helpers
- Bitmap helpers + hex encode/decode
- Logging helpers

# Architecture

## Modules

tracker/main.c

- Stores file table
- Stores peer list for each file with peer bitfields
- On ANNOUNCE updates peer bitfield
- On QUERY returns list of peers and meta

peer/main.c

- serveThread + serveOne: accept GET, read chunk from disk, send DATA
- announceToTracker: build local bitfield, send ANNOUNCE
- queryTracker: get peer list and meta from tracker
- worker threads + fetchChunk: download chunks and write to file
- Uses mutex and inFlightBitfield to avoid double work

common/proto.c + proto.h

- tConnectTcp, tListenTcp, tReadLine, tWriteAll
- tBitmapAlloc, tBitmapSet, tBitmapGet, tBitmapByteCount
- tHexEncode, tHexDecode
- tLog, tPerr

## Communication

- Peer -> Tracker: ANNOUNCE, QUERY
- - Peer -> Peer: GET, DATA

## Errors

Tracker:

- bad announce line -> ERR bad_announce
- file table full -> ERR file_table_full
- peer table full -> ERR peer_table_full
- meta mismatch -> ERR meta_mismatch
- bad bitmap -> ERR bad_bitmap
- memory issue -> ERR oom
- file not found on query -> ERR not_found
- unknown command -> ERR unknown_cmd

Peer (serve):

- bad request -> ERR bad_get

- missing file -> ERR no_file
- seek problem -> ERR seek
- memory problem -> ERR oom

Peer (get):

- if no peers -> query fails
- if chunk is missing on all peers -> stop and fail
- if peer returns wrong size -> mark that peer as not having this chunk
- inFlight bit is always cleared in cleanup to avoid deadlock

## Features

1. File is split into fixed size chunks (chunkSizeBytes)
2. Each peer sends a bitfield to tracker, so tracker knows which peer has which chunk
3. Downloader gets many peers from tracker using QUERY
4. Downloader uses worker threads (WORKERS) so it can download many chunks at same time
5. inFlightBitfield makes sure two workers do not download same chunk
6. pickSeedForChunk chooses a peer for a chunk (random start + scan), so work can spread between peers
7. If a peer lies or fails (wrong DATA size), downloader removes that chunk from this peer bitfield and tries others
8. Downloader writes each chunk to correct file offset, so chunks can arrive in any order
9. Resume support: if output file exists, downloader marks already-present chunks and continues
10. At end downloader prints summary how many chunks/bytes came from each peer, to see multi peer usage

# Network communications

Transport

- All communication uses TCP over IPv4 on a docker network.
- Tracker listens on 9000 (or TRACKER_PORT env)
- Each peer runs a TCP server on 10001, 10002, etc. (LISTEN_PORT env) to serve chunks to other peers.

Peer -> Tracker: ANNOUNCE

- Direction:
  - Peer (seeding node) -> Tracker
- When:
  - In "seed" mode (or periodic seeding loop), peer informs tracker what file/chunks it has.
- Request line format:
  - "ANNOUNCE <peer_port> <file_name> <full_size_bytes> <have_size_bytes> <chunk_size_bytes> <hex_bitfield>\n"
- Fields transferred:
  - peer_port: TCP port where this peer can serve GET requests.
  - file_name: logical file identifier.
  - full_size_bytes: total file size for this torrent (metadata).
  - have_size_bytes: local file size currently present on this peer (informational).
  - chunk_size_bytes: chunk size (e.g., 4096).
  - hex_bitfield: bitmap (hex-encoded) indicating which chunk indices the peer claims to have.
- Response:
  - "OK\n" on success
  - "ERR meta_mismatch\n" if announced size/chunk size differs from existing metadata
  - "ERR bad_bitmap\n" if bitmap length/format is invalid
  - other "ERR ...\n" on failure

Peer -> Tracker: QUERY

- Direction:
  - Peer (downloader) -> Tracker
- When:
  - At the start of "get" to retrieve torrent metadata + list of seed peers.
- Request line format:

    - QUERY <file_name>\n"
- Response header format:
    - "PEERS <file_name> <full_size_bytes> <chunk_size_bytes> <peer_count>\n"
- Response peer rows (repeated peer_count times):
    - "P <peer_ip> <peer_port> <hex_bitfield>\n"
- Fields transferred:
    - From tracker to peer: file metadata (full_size_bytes, chunk_size_bytes)
    - For each peer: address (peer_ip, peer_port) and availability bitmap (hex_bitfield)
- Error response:
    - "ERR not_found\n" if file unknown
    - "ERR bad_query\n" for malformed request

Peer -> Peer: GET

- Direction:
    - Downloader peer -> Seed peer
- When:
    - During "get" workers; for each missing chunk index the downloader connects to a selected seed.
- Request line format:
    - "GET <file_name> <chunk_index> <chunk_size_bytes>\n"
- (chunk_size_bytes is included to ensure both sides agree on read/offset size)
- Response:
    - Header: "DATA <nbytes>\n"
    - Body: raw bytes of the chunk (nbytes bytes)
- Notes on transferred information:
    - chunk_index determines file offset: chunk_index * chunk_size_bytes
    - nbytes is typically chunk_size_bytes, except for the final chunk where it may be smaller
- On invalid requests/conditions, seed replies with:
    - "ERR bad_get\n", "ERR no_file\n", "ERR seek\n", "ERR oom\n"

# Tests

1) Single seed

- One seed with full file, one downloader
- Expected: file downloads complete and matches size

2) Multi-seed partial files

- Seed A has first part of file, Seed B has later part
- Expected: downloader uses both peers and completes file

3) Meta mismatch

- Two peers announce same name but different total size or chunk size
- Expected: tracker returns ERR meta_mismatch for wrong announce

4) Bad bitmap

- Peer sends wrong hex string length or bad chars
- Expected: tracker returns ERR bad_bitmap

5) Missing chunk

- No peer has some chunk
- Expected: downloader stops and prints missing chunk info

6) Resume download

- Start download, stop it, run get again
- Expected: already have chunks are not downloaded again

7) Peer returns wrong DATA size

- Seed sends 0 bytes or too many bytes
- Expected: downloader rejects and tries another peer

# Demonstration Plan

1. Start docker containers, populate peer1 and peer2 with example file, peer1 will have it only partially, then announce files with “docker exec peer2 seed example.bin”, and same for peer1 but specify file size. Then download file with “get” command. Show the resulting file, compare with diff. Show docker logs
2. Run the test file, or show results of it running (as it takes time to run)
3. Show both on local PC and on BigUbu
4. Go through the code and explain how it works

# Running

Prerequisites

- Docker + Docker Compose plugin (docker compose)
- Bridge network on docker called “"z78_network” (or edit docker-compose.yaml)
- bash (or WSL on Windows) and python3 for the test scripts

Steps

1. Start the tracker + peers (docker)
    a. docker compose build
2. Start tracker and 3 peers
    a. docker compose up
3. Generate example files
    a. ./scripts/make_partials.sh
    b. Peer1 will have partial file
    c. Pee2 will have full file
4. seed from peer2
    a. docker exec t_peer1 app/tpeer seed example.bin
    b. save output file size
5. seed from peer1
    a. docker exec t_peer2 app/tpeer seed example.bin FILE_SIZE
6. download file to peer3
    a. docker exec t_peer3 app/tpeer get example.bin /data/example.bin

Logs

- tracker: docker logs t_tracker
- peers: docker logs t_peer1 / t_peer2 / t_peer3

Stop / cleanup

- docker compose down -v --remove-orphans

Tests

- ./tests/run-tests.sh

# Test results

Test downloading from one peer

```
[22:29:42] (peer:17) fetch: chunk=248 done nbytes=4096
[22:29:42] (peer:17) fetch: chunk=252 from 172.21.78.3:10001
[22:29:42] (peer:17) fetch: chunk=249 done nbytes=4096
[22:29:42] (peer:17) fetch: chunk=253 from 172.21.78.3:10001
[22:29:42] (peer:17) fetch: chunk=250 done nbytes=4096
[22:29:42] (peer:17) fetch: chunk=254 from 172.21.78.3:10001
[22:29:42] (peer:17) fetch: chunk=251 done nbytes=4096
[22:29:42] (peer:17) fetch: chunk=255 from 172.21.78.3:10001
[22:29:42] (peer:17) fetch: chunk=252 done nbytes=4096
[22:29:42] (peer:17) fetch: chunk=253 done nbytes=4096
[22:29:42] (peer:17) fetch: chunk=254 done nbytes=4096
[22:29:42] (peer:17) fetch: chunk=255 done nbytes=4096
[22:29:42] (peer:17) DOWNLOAD SOURCES SUMMARY (chunks/bytes by peer):
[22:29:42] (peer:17)   172.21.78.3:10001 -> chunks=256 bytes=1048576
[22:29:42] (peer:17) GET complete file=example.bin out=/data/out.bin
stdin: is not a tty
stdin: is not a tty
stdin: is not a tty
testHappyPathSingleSeed OK
```

Test downloading from multiple peers

```
[22:30:25] (peer:17) fetch: chunk=251 done nbytes=4096
[22:30:25] (peer:17) fetch: chunk=255 from 172.21.78.5:10001
[22:30:25] (peer:17) fetch: chunk=252 done nbytes=4096
[22:30:25] (peer:17) fetch: chunk=253 done nbytes=4096
[22:30:25] (peer:17) fetch: chunk=254 done nbytes=4096
[22:30:25] (peer:17) fetch: chunk=255 done nbytes=4096
[22:30:25] (peer:17) DOWNLOAD SOURCES SUMMARY (chunks/bytes by peer):
[22:30:25] (peer:17)   172.21.78.5:10001 -> chunks=215 bytes=880640
[22:30:25] (peer:17)   172.21.78.4:10002 -> chunks=41 bytes=167936
[22:30:25] (peer:17) GET complete file=example.bin out=/data/out.bin
stdin: is not a tty
stdin: is not a tty
stdin: is not a tty
testHappyPathMultiSeed OK
```

Test that we get “meta mismatch” when seeding two files but different file sizes

```
dockerExec t_peer1 sh -lc "/app/tpeer seed example.bin 1048576" >/dev/null
dockerExec t_peer2 sh -lc "/app/tpeer seed example.bin 999999" >/dev/null 2>&1 || true
```

```
== testMetaMismatch ==

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 4/4
 ⠿ Container t_tracker  Started 1.4ss
 ⠿ Container t_peer2    Started 2.0ss
 ⠿ Container t_peer1    Started 2.6ss
 ⠿ Container t_peer3    Started 2.5ss
stdin: is not a tty
[22:31:05] (peer:17) ANNOUNCE ok file=example.bin full_size=1048576 have_size=1048576 chunk=4096
testMetaMismatch OK
```

Test incorrect ANNOUNCE

```
docker run --rm --network psinet alpine:3.19 sh -lc "
  apk add --no-cache netcat-openbsd >/dev/null
  printf 'ANNOUNCE 5555 example.bin 1048576 1048576 4096 deadbeef\n' | nc -w 2 t_tracker 9000
" > "$ROOT_DIR/tests/_badBitmap.out" 2>&1 || true
```

```
1   tracker listening on :9000
2   [22:31:42] (tracker:1) client 172.21.78.6 connected
3   [22:31:42] (tracker:1) ANNOUNCE from 172.21.78.6:5555 file=example.bin: bad bitmap
```

```
== testBadBitmap ==

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 4/4
 ⠿ Container t_tracker  Started 0.8ss
 ⠿ Container t_peer2    Started 1.6ss
 ⠿ Container t_peer3    Started 2.0ss
 ⠿ Container t_peer1    Started 1.9ss
testBadBitmap OK
```

Test no chunk at no peers

```
  stdin: is not a tty
  [22:32:23] (peer:17) GET start file=example.bin out=/data/out.bin tracker=t_tracker:9000
  [22:32:23] (peer:17) seed[0]=172.21.78.5:10001
  [22:32:23] (peer:17) seed[1]=172.21.78.4:10002
  [22:32:23] (peer:17) QUERY file=example.bin → size=1048576 chunk=4096 nchunks=256 npeers=2
  [22:32:23] (peer:17) GET failed: missing chunk 0 (no peer has it)
  [22:32:23] (peer:17) GET failed: missing chunk 0 (no peer has it)
  [22:32:23] (peer:17) GET failed: missing chunk 0 (no peer has it)
  [22:32:23] (peer:17) GET failed: missing chunk 0 (no peer has it)
  [22:32:23] (peer:17) DOWNLOAD SOURCES SUMMARY (chunks/bytes by peer):
  [22:32:23] (peer:17) GET incomplete file=example.bin out=/data/out.bin
```

```
  [+] Running 4/4
   ⠿ Container t_tracker  Started 0.7ss
   ⠿ Container t_peer1    Started 1.9ss
   ⠿ Container t_peer2    Started 1.9ss
   ⠿ Container t_peer3    Started 1.8ss
  stdin: is not a tty
  [22:32:21] (peer:17) ANNOUNCE ok file=example.bin full_size=1048576 have_size=1
  hunk=4096
  stdin: is not a tty
  [22:32:22] (peer:17) ANNOUNCE ok file=example.bin full_size=1048576 have_size=1
  hunk=4096
  testMissingChunk OK
```

Test resuming download (Not aligned to chunk size, checking hashes)

```
dockerExec t_peer3 sh -lc "rm -f /data/out.bin"
dockerExec t_peer3 sh -lc "dd if=/data/example.bin of=/data/out.bin bs=1 count=200000 status=none" >/dev/null
```

```
 ⠿ Container t_tracker  Started 0.7ss
 ⠿ Container t_peer3    Started 1.7ss
 ⠿ Container t_peer1    Started 1.7ss
 ⠿ Container t_peer2    Started 1.7ss
stdin: is not a tty
[22:32:56] (peer:16) ANNOUNCE ok file=example.bin full_size=1048576 have_size=1048576 chunk=4096
stdin: is not a tty
[22:32:56] (peer:18) ANNOUNCE ok file=example.bin full_size=1048576 have_size=400000 chunk=4096
stdin: is not a tty
stdin: is not a tty
stdin: is not a tty
stdin: is not a tty
stdin: is not a tty
testResumeDownload OK
```

Test with fake seeder that announces it has chunk but gives chunk with bad size

(longest test but passes)

```
[22:37:24] (peer:17) fetch: chunk=255 from 172.21.78.5:10001
[22:37:24] (peer:17) fetch: chunk=254 done nbytes=4096
[22:37:24] (peer:17) fetch: chunk=255 done nbytes=4096
[22:37:27] (peer:17) connect seed: No route to host (errno=113)
[22:37:27] (peer:17) [22:37:27] (peer:17) connect seed: No route to host (errno=113)
fetch: chunk=246 from 172.21.78.5:10001[22:37:27] (peer:17) fetch: chunk=251 from 172.21.78.5:10001

[22:37:27] (peer:17) fetch: chunk=251 done nbytes=4096
[22:37:27] (peer:17) fetch: chunk=246 done nbytes=4096
[22:37:27] (peer:17) DOWNLOAD SOURCES SUMMARY (chunks/bytes by peer):
[22:37:27] (peer:17)   172.21.78.5:10001 -> chunks=256 bytes=1048576
[22:37:27] (peer:17) GET complete file=example.bin out=/data/out.bin
stdin: is not a tty
stdin: is not a tty
stdin: is not a tty
testBadDataSizeFromSeed OK
```

Testing trying to get a file which is not announced (not found)

tests > _logs > peer3_notFound.log

```
1  stdin: is not a tty
2  [22:38:10] (peer:16) GET start file=missing.bin out=/data/out.bin tracker=t_tracker:9000
3  [22:38:10] (peer:16) QUERY unexpected: ERR not_found
4
5  [22:38:10] (peer:16) GET: query failed
6
```

```
== testNotFoundQuery ==

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 4/4
 ⠿ Container t_tracker  Started 1.6ss
 ⠿ Container t_peer2    Started 2.8ss
 ⠿ Container t_peer3    Started 2.9ss
 ⠿ Container t_peer1    Started 2.8ss
testNotFoundQuery OK
ALL TESTS OK
```