دانشگاه صنعتی شریف

شبکه‌های کامپیوتری
شبکه تورنت

مائده حیدری | ۴۰۰۱۰۴۹۱۸

پاییز ۱۴۰۳

1. Overview of the Project

هدف این پروژه، پیاده‌سازی یک شبکه‌ی ساده **Torrent** با ویژگی‌های زیر است:

1. **یک Tracker** که اطلاعات فایل‌ها و Peerها را مدیریت می‌کند.
2. **حداقل سه Peer** برای اشتراک‌گذاری و دانلود فایل‌ها.
3. استفاده از **UDP** برای ارتباط بین Tracker و Peerها.
4. انتقال داده‌های فایل از طریق **TCP** بین Peerها.
5. وجود یک سیستم **Logging** هم در Tracker و هم در Peerها.
6. مکانیزمی برای اطلاع‌یافتن Tracker از آنلاین یا آفلاین‌بودن Peerها تا Peerهای ازکارافتاده به دیگران معرفی نشوند.

2. Implemented Features

در این بخش به ویژگی‌های اصلی پیاده‌سازی‌شده اشاره می‌شود:

1. **Multi-Instance**
   - امکان اجرای یک Tracker و چندین Peer به‌صورت هم‌زمان، مطابق نیاز پروژه برای حداقل سه Peer.
2. **Tracker Capabilities**
   - **نگهداری** وضعیت این‌که کدام Peer چه فایل‌هایی را کامل در اختیار دارد.
   - **پاسخ** به درخواست‌های دریافت فایل شامل اندازه فایل و فهرست تمام Seederها.
   - **ثبت رخدادها** (Log) برای تمام فعالیت‌ها (مانند SHARE یا GET).
   - **ارائهٔ محیط interactive برای اجرای دستورات:**
     - request logs جهت مشاهدهٔ فهرست درخواست‌ها
     - all-logs جهت مشاهدهٔ وضعیت تمام فایل‌ها و Peerهای دارندهٔ آن‌ها
     - <filename> file_logs> جهت مشاهدهٔ اطلاعات مربوط به یک فایل خاص
     - exit جهت پایان اجرای Tracker
3. **Peer Capabilities**
   - Peerها در یکی از دو حالت **share** یا **get** اجرا می‌شوند.

- در حالت **share**، یک فایل را به Tracker اعلام (SHARE) می‌کنند تا به‌عنوان Seeder شناخته شوند.
- در حالت **get**، با ارسال درخواست به Tracker، لیست Seederها را دریافت کرده و فایل را از یکی از آن‌ها (به‌صورت تصادفی) دانلود می‌کنند. پس از اتمام دانلود، Peer به Seeder تبدیل می‌شود.
- کامندها برای هر Peer:
  - request logs جهت مشاهدهٔ گزارش‌های محلی در Peer
  - exit جهت خروج و ارسال پیام DISCONNECT به Tracker

4. **Heartbeat/Disconnection Mechanism**
   - پیاده‌سازی سیگنال‌های **Heartbeat** تا Peerها به شکل دوره‌ای وضعیت فعال‌بودن خود را به Tracker اطلاع دهند.
   - اگر Tracker طی مهلت تعیین‌شده (Timeout) پیامی از Peer دریافت نکند، آن Peer را آفلاین فرض کرده و از فهرست Seederها حذف می‌کند.
   - هر Peer در صورت خروج عادی، با ارسال پیام DISCONNECT به Tracker، خود را از سیستم خارج می‌کند.
5. **Multithreading**
   - **Tracker**:
     - یک Thread برای دریافت پیام‌های UDP
     - یک Thread برای بررسی Heartbeat Peerها
     - Thread اصلی برای پردازش دستورات در کنسول
   - **Peer**:
     - یک Thread جهت ارسال Heartbeat
     - یک Thread برای راه‌اندازی سرور TCP و پاسخ‌گویی به درخواست فایل از سایر Peerها
     - Thread اصلی برای دریافت دستورات کاربر، همچنین مدیریت دانلود فایل
6. **Logging System**
   - **Tracker** تمام درخواست‌های ورودی (REGISTER، SHARE، GET، DISCONNECT و ...) را ذخیره می‌کند.
   - **Peer** نیز هر پیام ارسالی به Tracker یا دریافت فایل و آپلود را ثبت می‌کند تا بتواند از طریق request logs به آن‌ها دسترسی داشته باشد.

3. Explanation of the Code Structure

3.1 Tracker Program

Main Components

- **کلاس Tracker** که از files_dict برای نگهداری فایل‌ها و peer_info برای اطلاعات Peerها بهره می‌برد.
- متد listen_for_peers که در یک Thread به‌طور مداوم پیام‌های UDP ورودی را دریافت و به handle_message منتقل می‌کند.
- متد monitor_peers که به شکل منظم زمان آخرین Heartbeat را برای هر Peer بررسی و در صورت Timeout، آن Peer را حذف می‌کند.
- متدهای کلیدی handle_success_download ،handle_get_request ،share_file ،register_peer و handle_disconnect. هرکدام وظیفهٔ خاصی مانند ثبت Peer جدید یا حذف Peer ازکارافتاده را بر عهده دارند.
- پشتیبانی از کامندها جهت مشاهدهٔ گزارش‌ها (request logs و غیره).

(Execution Flow (Tracker

اجرای Tracker با دستور:

python tracker.py 127.0.0.1:6771

1. Tracker در حال دریافت پیام‌های UDP از Peerها خواهد بود.
2. Thread مانیتورینگ، Peerهایی را که Heartbeat ارسال نکنند، حذف می‌کند.
3. کاربر می‌تواند با کامندها (request logs, all-logs, file_logs <filename>, exit) وضعیت را بررسی یا برنامه را خاتمه دهد.

3.2 Peer Program

Main Components

- **کلاس Peer** که در آن مشخص می‌شود Peer در حالت share یا get اجرا شده است، چه نام فایلی را اعلام می‌کند یا دانلود می‌کند، و آدرس‌های Tracker و TCP چیست.
- متد share_file که اعلان (SHARE) را برای Tracker ارسال می‌کند تا این فایل در اختیار این Peer شناخته شود.

- متد get_file که با ارسال (GET) از Tracker لیست Seederها را می‌گیرد و سپس متد download_file_from را برای دانلود فایل از یک Seeder خاص فرامی‌خواند.
- متد handle_incoming_connections که در یک Thread در حال گوش‌دادن (Listen) روی TCP server است تا اگر Peer دیگری درخواست فایل داشت، آن را آپلود کند.
- متد disconnect_from_tracker برای زمانی که کاربر با فرمان exit قصد خروج دارد و می‌خواهد پیام DISCONNECT به Tracker بفرستد.
- یک Thread برای ارسال سیگنال **Heartbeat**، تا Tracker بداند این Peer همچنان آنلاین است.

(Execution Flow (Peer

به‌عنوان مثال، اجرای Peer در حالت Share:

python peer.py share myfile.txt 127.0.0.1:6771 127.0.0.1:7001

1.
   - این Peer پس از ثبت‌نام (REGISTER)، سرور TCP خود را راه‌اندازی می‌کند و با فرستادن SHARE، اعلام می‌کند که فایل myfile.txt را در اختیار دارد.
   - هم‌زمان در فواصل زمانی تعیین‌شده، Heartbeat به Tracker ارسال می‌شود.

اگر Peer در حالت Get باشد:

python peer.py get myfile.txt 127.0.0.1:6771 127.0.0.1:7002

2.
   - پیام GET را می‌فرستد تا لیست Seederها را دریافت کند، سپس از یکی از آن‌ها از طریق TCP دانلود می‌کند. وقتی دانلود تمام شود، پیام SUCCESS_DOWNLOAD به Tracker فرستاده و تبدیل به Seeder می‌شود.
3. در نهایت با فرمان exit، پیام DISCONNECT ارسال شده و Peer به‌شکل رسمی از شبکه خارج می‌شود.

4. Steps of Code Execution

نمونه سناریو:

راه‌اندازی Tracker

python tracker.py 127.0.0.1:6771

1. Tracker در پورت UDP مربوطه منتظر خواهد ماند.

Peer A در حالت Share

python peer.py share myfile.txt 127.0.0.1:6771 127.0.0.1:7001

.2

- Peer A اعلام می‌کند که فایل myfile.txt را کامل دارد و شروع به ارسال Heartbeat می‌کند.

Peer B در حالت Get

python peer.py get myfile.txt 127.0.0.1:6771 127.0.0.1:7002

.3

- Tracker بیان می‌کند که Peer A فایل را دارد.
- Peer B از طریق TCP فایل را از Peer A می‌گیرد و در پایان به Seeder تبدیل می‌شود.

Peer C در حالت Get

python peer.py get myfile.txt 127.0.0.1:6771 127.0.0.1:7003

.4

- اکنون Tracker می‌داند که هم Peer A و هم Peer B فایل را دارند. Peer C تصادفی یکی را انتخاب می‌کند و فایل را می‌گیرد.

5. **Heartbeat & Inactivity**
   - همهٔ Peerها با فواصل منظم Heartbeat می‌فرستند. درصورت قطع شدن ناگهانی یک Peer، Tracker پس از سپری شدن زمان Timeout او را حذف می‌کند.
6. **بررسی لاگ‌ها**
   - در Tracker: request logs یا all-logs یا file_logs a.txt برای گزارش کلی یا فایل مشخص.
   - در هر Peer: request logs برای مشاهدهٔ رویدادهای محلی.
7. **خروج**
   - با exit در Peer، پیام DISCONNECT فرستاده می‌شود و Peer از Tracker خارج می‌گردد. همچنین تایپ exit در Tracker، برنامه را به طور کلی خاتمه می‌دهد.

در ادامه محتوای فایل test.txt قابل مشاهده است که حاوی دستورات اجرا شده در فیلم گزارش است.

```
# Create Test Files
echo "This is test file 1" > file1.txt
echo "This is test file 2" > file2.txt
echo "This is test file 3" > file3.txt


# Start tracker (in terminal 1)
python3 tracker.py 127.0.0.1:6771

# Peer Operations

# Start Peer1 (Sharing file1.txt) - in terminal 2
mkdir -p peer1_dir
cp file1.txt peer1_dir/
cd peer1_dir
python3 ../peer.py share file1.txt 127.0.0.1:6771 127.0.0.1:7001 Peer1
cd ..

# Start Peer2 (Sharing file2.txt) - in terminal 3
mkdir -p peer2_dir
cp file2.txt peer2_dir/
cd peer2_dir
python3 ../peer.py share file2.txt 127.0.0.1:6771 127.0.0.1:7002 Peer2
cd ..

# Start Peer3 (Downloading file1.txt) - in terminal 4
mkdir -p peer3_dir
cd peer3_dir
python3 ../peer.py get file1.txt 127.0.0.1:6771 127.0.0.1:7003 Peer3_1
cd ..

# Test Scenarios

# Test Scenario 1: Peer3 downloading file1.txt from Peer1
# Already covered by starting Peer3

# Test Scenario 2: Peer3 trying to download a non-existent file - in
terminal 5
cd peer3_dir
python3 ../peer.py get nonexistent.txt 127.0.0.1:6771 127.0.0.1:7013
Peer3_2
cd ..

# Test Scenario 3: Peer3 downloading file2.txt from Peer2 - in terminal 6
```

```
cd peer3_dir
python3 ../peer.py get file2.txt 127.0.0.1:6771 127.0.0.1:7023 Peer3_3
cd ..

# Test Scenario 4: Peer1 disconnecting
pkill -f "python3.*Peer1"
sleep 2

# Test Scenario 5: New download of file1.txt after Peer1 disconnected -
in terminal 7
mkdir -p peer4_dir
cd peer4_dir
python3 ../peer.py get file1.txt 127.0.0.1:6771 127.0.0.1:7004 Peer4
cd ..

# Tracker commands
request logs
all-logs
file_logs file1.txt
file_logs file2.txt
```