بنام خدا
اول از همه یک نکته کلید را بگم که چون من فقط روی یک ماشین تست کردم پس ناچاراً 2 یا چندتا نود داشتم از یک نرم‌افزار پس یک ستون هم به ستون‌های لیست نودها اضافه کردم که پورت UDP را مشخص می‌کند برای هر نود. اگر نرم‌افزار روی شبکه اینترنت می‌خواست کار کنه نیازی نبود و یک قراردادی می‌کردیم روی شماره پورت UDP و همون رو برای همه قرار می دادیم ولی اینجا نمی‌شود ولی همانطور در مورد کانکشن TCP در متن پروژه گفته شده است،پورت این کانکشن در پاسخ GET مشخص می‌شود که روی کدام برقرار شود و بصورت تصادفی در زمان ران شدن نود تعیین می‌شود.

راهنمای اجرای برنامه p2p

سورس کد را بصورت یک jar اکسپورت کنید و یا اگر می‌خواهید می‌توانید از فایل jar موجود در پوشه out روی ریپازیتوری استفاده کنید.

فایل jar را داخل یک پوشه بریزید
ساختار دایرکتوری باید به شکل زیر باشه :

```
├── data
│   ├── config.p2p
│   ├── f
│   │   ├── muzik.mp3
│   │   └── pic.jpg
│   └── p2n.txt
└── p2p.jar
```

که پوشه data الزامی است که کنار فایل jar باشد.
همه آدرس دهی ها بر مبنای این پوشه data خواهد بود.

به مسیری که فایل jar را ریخته‌ایم می‌رویم و این کامند را ران میکنیم:

```
java -jar p2p.jar
```

با این کار یک نود از شبکه بالا می‌آید.
برای دیدین مسیج های دیباگ از دستور زیر استفاده کنید که تمامی مسیج های دیباگ روی کنسول نمایش داده می شوند.
این دستور نود را برای حالت دیباگ بالا می آورد:

```
java -jar p2p.jar debug
```

وقتی برنامه ران شد یک سری سؤال از شما می‌پرسد که مسیر دهی ها باید بر مبنای پوشه data باشد و اینکه توجه کنید اگر میگویید مثلاً p2n.txt فایل نود های من است باید آن فایل قبلاً در پوشه data ساخته شده باشد.
همچنین دایرکتوریی که مخزن فایل‌ها است باید در پوشه data ساخته شده باشد. مثل پوشه f که اکنون شامل یک فایل صوتی و یک فایل عکس است.
این کانفیگ ها برای یکبار که اولین بار اجرا می‌شود انجام می‌شود و در اجرا های بعدی از فایل کانفیگ لود می‌کند و نیازی به وارد کردن دوباره کانفیگ ها نیست.

سواری مجانی:
برای این کار ابتدا چک می‌شود که قبلن به این نود سرویس داده‌ایم یا نه اگر نداده باشیم بلافاصله جواب GET را با پکت UDP می دهد. در صورتی که این نود از ما درخواست فایل کند حالا یکی به ضریب delay اضافه می‌کنیم که هر چه بیشتر باشد بیشتر منتظر پاسخ GET در درخواست های بعدی خواهد بود.
برای کم کردن این ضریب تأخیر وقتی یک فایل به صورت موفقیت آمیز دریافت شود یکی از ضریب نود ارسال کننده کم می کنیم.

یک نکته دیگر هم در این بخش لحاظ شده است آن هم اینکه هر نودی که از ما درخواست فایلی می‌کند اگر در لیست نودهای ما نباشد پاسخ نمی‌گیرد دلیل آن هم این است که وقتی ما آن را در لیست خود نداریم یعنی خودش را به ما معرفی نکرده است ... که یا جدید است یا عمداً معرفی نمی‌کند که ما درخواست GET برای آن نفرستیم!!!

بسته‌های UDP:
در مورد پکت های UDP اینکه سایز پکت ها ماکزیمم به اندازه 62 کیلوبایت ست شده است که در بخش ثابت ها (Constants) می‌توان تغییرش داد(همه محاسبات و پیام‌های سیستم بر اساس این مقدار است پس نگران تغییر آن نباشید ... همه جا تغییر می کند)
در مورد ساختار پیام‌های UDP اینگونه است که 4 بایت اول اندازه payload و 4 بایت بعدی تایپ پیام را نشان می‌دهد که در بخش ثابت ها می‌توانید شماره هر تایپ را مشاهده کنید.
در مورد اینکه چرا 32 بیتی گرفتیم علت آنکه چون راحت‌تر است که یک int را بریزی توش و برداری وگرنه برای تایپ مثلاً کلاً نیاز به 2 بیت است (3 تایپ پیام داریم)
ولی کار در حد بیت و بایت در جاوا کمی اذیت کننده است.(با این حال نمی‌گوییم نمی‌شود چون هدف پروژه تمرکز روی این بخش نبود ما هم از این بهینه سازی صرف نظر کردیم.)
در ادامه این 8 بایت payload شروع می شود.
Payload دراصل بایت های آبجکت های جاوایی است که به اصلاح آن را DTO (Data Transfer Object) می‌نامیم (که در جاوا بسیار مرسوم است بخصوص بحث شبکه و وب)
این کلاس‌ها لاجیکی ندارند و فقط دیتا در آن‌ها ریخته می‌شود سپس به کمک رویش هایی بایت های این دیتا در می‌آید و در در ادامه آن 8 بیت پر می‌شود و در سمت دیگر بر اساس تایپ پیام بایت ها تبدیل به یک DTO می‌شوند. (از هیچ کتاب خانه ی خارجی یا آماده‌ای برای این کار استفاده نشده است و بصورت دستی و به کمک آی/اُ استریم های جاوا نوشته شده است.)

انتخاب بهترین پاسخ:
وقتی ریکوئست GET را می‌خواهیم جواب دهیم مقدار زمان کنونی را هم می فرستیم و در سمت گیرنده (فرستنده GET) به محض دریافت پاسخ تأخیر آمدن را با توجه با زمان کنونی محسابه می‌کنیم و نگه می‌دارم سپس که تایمر زمان انتظار تمام شد بهترین پاسخ بر اساس کمترین تأخیر انتخاب می‌شود و به مرحله بعدی داده می‌شود.

برای تست:
یک یا چند نود بصورت دیباگ در تب های متفاوت کنسول بالا آورده شود و سپس یک نود هم بصورت عادی بالا آورده شود و با این نود عادی می‌توانید ریکوئست بدهید و با نود های دیگر هم میتوانید تعامل بین نودها و پاسخ‌ها به نود عادی را مشاهده کنید.

نکته مهم اینکه در صورتی که بخواهید اپ را با مود debug بالا آورده و بصورت interactive با آن کار کنید به مشکل می‌خورید زیرا پیام‌های زیادی روی کنسول می نویسد. در این حالت برای تست باید کامند را به همراه یک خط خالی در فایل تکست نوشته و پس از کپی در ترمنال کپی کنید تا بلافاصله بعد از کیپ کامند ران شود و شما نیز لاگ را ببینید (برای چک کردن اینکه آیا بهترین پاسخ را انتخاب می‌کند یا نه باید اینگونه تست شود)

درآخر:
کد ها و کامیت های پروژه در ریپازیتوری زیر موجود می‌باشد که سعی میکنم یک‌شنبه شب پابلیک کنم.
https://gitlab.com/alifa98/p2p

با تشکر- علی فرجی