دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش پروژه اول درس شبکه‌های کامپیوتری

پیاده‌سازی یک پیام‌رسان ساده

استاد درس
دکتر حجازی

نگارش
دانیال حمدی

پاییز ۹۹

۱. فرضیات پروژه

۱.۱. توضیحات درمورد پروژه

در این پروژه قصد داریم یک پیام‌رسان مشابه پیام‌رسان‌های امروزی پیاده‌سازی نماییم. همانطور که می‌دانید پیامرسانهای امروزی علاوه بر تبادل متن و فایل، قابلیت تماس صوتی و تصویری، ارسال استیکر، ایجاد گروه و ... دارند که در این پروژه تمرکز بر روی قابلیت ارسال متن و فایل، ایجاد گروه (امتیازی) و تغییر نام کاربر میباشد.

پیاده سازی میتواند به صورت رابط گرافیکی ((GUIیا با استفاده از رابط خط فرمان ((CLIباشد. روش پیادهسازی بدین صورت است که ابتدا می‌بایست یک سرور ایجاد و اجرا شود و کاربران با یک نام کاربری یکتا به آن متصل شوند (هیچ دو کاربری نباید نام یکسان داشته باشند). هر کاربر دارای 3 مقدار در جدول مسیریابی سرور میباشد: نام کاربری، IP که کاربر به سرور متصل می‌شود و پورت اتصالی به سرور. هنگامی که کاربری به سرور متصل می‌شود، مقادیر مربوطه آن در جدول مسیریابی درون سرور ثبت شده و می‌بایست به سایر کاربران اطلاع داده شود که که کاربر جدید اضافه شده است و کاربران می‌توانند با کاربر جدید تبادل پیام و فایل انجام دهند. هنگامی که کاربر 1 قصد ارسال پیام یا فایل به کاربر 2 را دارد، باید به سرور درخواست اتصال به کاربر 2 را بدهد. سرور در جدول اطلاعات ذخیره‌شده خود کاربر2 را جستجو نموده، اگر موجود بود به کاربر1 اطلاع داده و به کاربر 2 نیز متصل شود و پس از آن کاربر 1 پیام‌های خود را از طریق سرور به کاربر 2 ارسال می‌کند. پیام می‌تواند به صورت متن یا فایل باشد. برای تبادل فایل می‌بایست فایل را به بسته‌هایی با حجم مشخص تکه‌تکه نموده و هر بسته را جدا جدا ارسال نموده و در مقصد آن را یکپارچه کند. برای تغییر نام کاربر نیز نام کاربری جدید نباید با هیچ‌یک از سایر کاربران متصل به سرور یکسان باشد و در صورت افتادن این اتفاق می‌بایست پیغام خطای مناسب نشان داده شود.

۲.۱. نکات پیاده‌سازی

- ذخیره‌سازی اطلاعات کاربران در سرور می‌تواند در یک فایل متنی یا داخل یک پایگاه داده باشد.
- سرور باید به صورت multi-thread عمل کند تا قابلیت ارتباط هم‌زمان با چندین client را داشته باشد.
- هر کاربر می‌تواند لیست تمام کاربرانی که امکان پیام دادن به آنها را دارد مشاهده کند.
- سرور می‌تواند در هر لحظه لیست تمام کاربران را مشاهده کند.

- سرور ورود کاربران به سیستم و خروج آنها را به اطلاع سایر کاربران می‌رساند.
- زبان پیاده‌سازی پروژه آزاد است.
- ارسال و دریافت بسته‌ها با استفاده از سوکت صورت می‌گیرد و قسمت‌های مربوط به ساخت، گوش دادن، ارسال و دریافت می‌بایست در کد قابل رویت باشند.

۲. شرح گزارش

۱.۲. مشخص کردن شرکت‌کننده‌های سیستم و رسم FSM

برنامه از دو ماژول کلاینت و سرور تشکیل شده است. ماژول سرور به دو کلاس Server و ServerSocket تقسیم شده است. همچنین ماژول کلاینت از یک کلاس Client تشکیل شده که این کلاس با دو کلاس SendThread و ReceiveThread کار می‌کند. توضیح هر کدام از این کلاس‌ها را در قسمت بعد می‌آوریم.

کلاس‌های برنامه stateهای زیادی ندارند، به همین دلیل نمودار FSM آن ساده است.

کلاس سرور در ابتدا در حالت بی‌کار (idle) می‌باشد، با فراخوانی متد start()، شروع به کار و گوش دادن به درخواست‌های جدید می‌کند. مسئول سرور می‌تواند با فشردن کلید q سرور را به حالت idle بازگرداند.

کلاس کلاینت هم در ابتدا idle است و با فرخوانی متد start() شروع به کار می‌کند. در لحظه از زمان، کاربر می‌تواند با وارد کردن /quit از برنامه خارج شود.

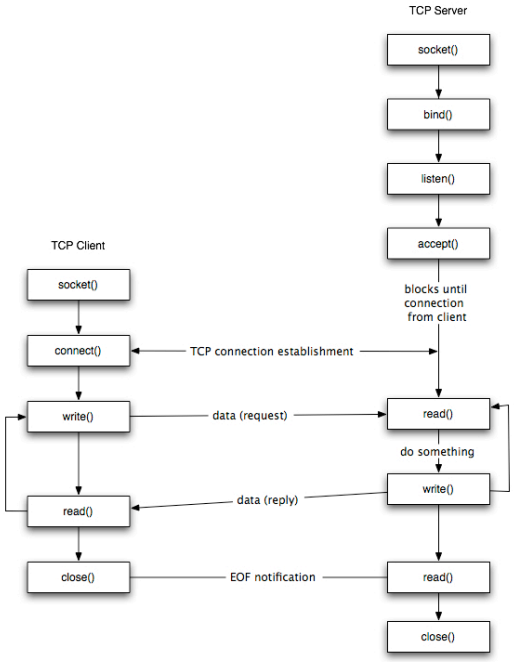
۲.۲. توضیح معماری پیام‌رسان و ارتباط میان توابع

در شکل زیر معماری کلی یک ارتباط کلاینت-سرور با استفاده از سوکت قابل مشاهده است.

به طور کلی روند برنامه به این گونه است که یک سوکت در سمت سرور می‌سازیم (socket())، برای آن یک آدرس آی‌پی و پورت مشخص می‌کنیم (bind()) تا به درخواست‌ها به این آدرس گوش دهد. (listen())

از این پس، هر کلاینت می‌تواند یک سوکت بسازد (socket()) و به سوکت سرور درخواست ارتباط (connect()) دهد.

سوکت این ارتباط را قبول می‌کند (accept()) و ارتباط شکل می‌گیرد.



از آن جایی که هر سرور باید با چندین کلاینت در ارتباط باشد، باید سرور را به صورت Multithread پیاده‌سازی کنیم؛ به این صورت که برای هر کلاینت متصل به سرور، یک thread جداگانه (شامل یک سوکت) در حال اجرا باشد. این سوکت به طور پیوسته آماده‌ی دریافت و ارسال پیام به کلاینت نظیر خودش است. در کد برای سوکت‌های نظیر هر کلاینت، از کلاس ServerSocket استفاده شده است.

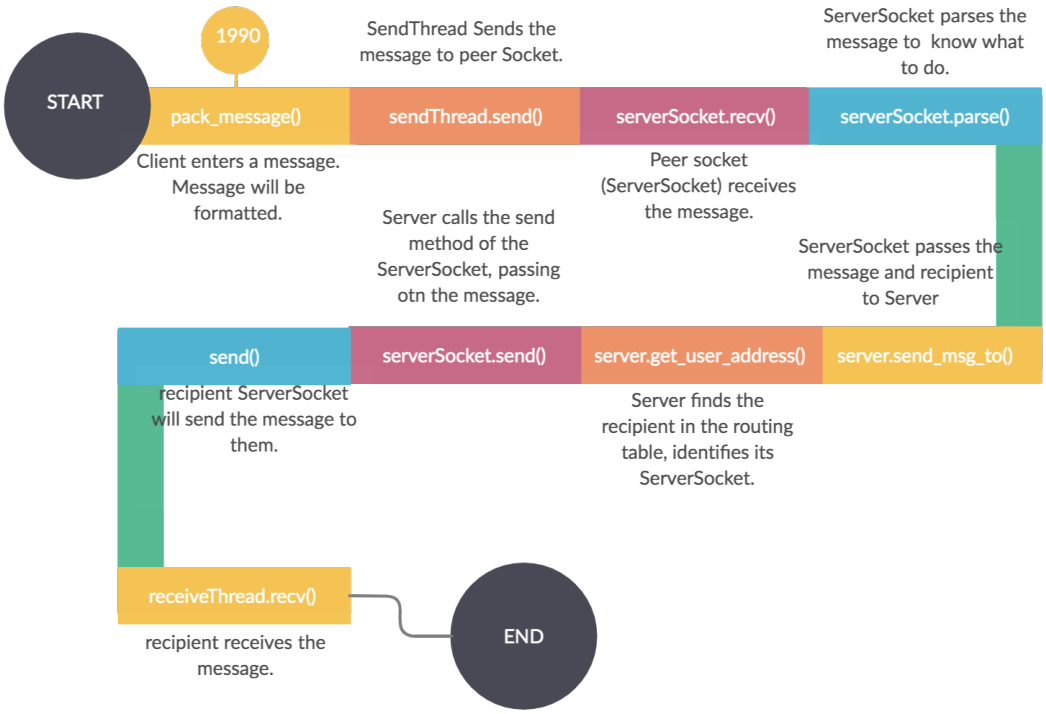
بنابراین هر کلاینت پیوسته می‌تواند ارسال پیام (send()) و یا دریافت پیام (recv()) از این سوکت نظیرش داشته باشد.

در نتیجه به طور کلی برنامه از سه کلاس اصلی Client, Server, ServerSocket تشکیل شده است.

۳.۲. توضیح توابع پیاده‌سازی شده

تمام توابع و کلاس‌های برنامه دارای مستندسازی (doc) و کامنت می‌باشند؛ بنابراین برای اطلاع از کاربرد هر تابع می‌توان به داک آن رجوع کرد.

در ادامه جریان اجرای برنامه و توابع فراخوانی شده طی ارسال و دریافت یک پیام از یک کاربر به کاربر دیگر را بررسی می‌کنیم.



همان‌طور که در شکل پیداست، کاربر ابتدا پیام خود را وارد می‌کند. این پیام به فرمت توسط متد pack_message() به فرمت زیر درمی‌آید:

“From: `sender`|To: `recipient`|Content: `this is an example`”

پس از این، با استفاده از متد send کلاس SendThread، این پیام به سوکت نظیر کاربر در سمت سرور ارسال می‌شود.

سوکت کلاس ServerSocket این پیام را با متد recv() دریافت می‌کند. سپس با استفاده از متد parse() آن را بررسی می‌کند. این پیام می‌تواند یک دستور (مثلا ایجاد یک گروه) یا یک پیغام به یک کاربر باشد. متد parse() این موارد را تشخیص می‌دهد و سپس متد مناسب را بسته به تشخیصش صدا می‌زند.

برای مثال در شکل بالا، متد parse تشخیص داده که بسته‌ی دریافت شده، به یک کاربر خاص است، پس متد ارسال پیام را از سرور صدا می‌زند و محتوا و مخاطب پیام را به آن پاس می‌دهد. (send_msg_to())

سرور ابتدا از جدول مسیریابی آدرس مخاطب را پیدا می‌کند، سپس طبق آن آدرس، آبجکت serverSocket نظیر آن مخاطب را پیدا می‌کند و پیام را به سوکت نظیر پاس می‌دهد تا به مخاطب ارسال کند.

در نتیجه کلاس serverSocket با متد send این پیغام را به مخاطب ارسال می‌کند.

در نهایت receiveThread در سمت کلاینت پیغام را دریافت، و به کاربر نشان می‌دهد.

در سناریو بالا، از یک نکته عبور کردیم و آن هم این که سمت کلاینت هم باید از دو thread موازی تشکیل شده باشد؛ چرا که هر کلاینت به طور پیوسته هم در حال ارسال، و هم در حال دریافت پیام می‌باشد. بنابراین برای هر کلاینت دو thread موازی SendThread و ReceiveThread پیاده‌سازی می‌کنیم.

پایان