بنام خدا

پروژه برنامه نویسی درس شبکه های کامپیوتری

Socket Programming

استاد درس: دکتر مرتضی آنالویی

کمک مدرس: پیام مهاجری

میلاد تیموری

۹۱۵۲۲۰۵۹

پرسش های تحلیلی

1) مشکلات موجود در طراحی پروتکل ارتباطی این پروژه را تحلیل کرده و راهکار خود را برای آن پیشنهاد دهید:

شاید بزرگترین چالش در طراحی این پروتکل جلوگیری از ایجاد race بین نخ های برنامه ها باشد، که برای جلوگیری از این مشکل در برنامه سمت کاربر برای گوش کردن سوکت، تنها یک Thread ایجاد شده و ارسال دستورات در همان main برنامه انجام میشود، در سمت سرور نیز با ساخت یک Thread برای هر کاربر و سراسری (global) اعلام کردن متغیر های مناسب از این مشکل در این برنامه جلوگیری شده است.
همچنین یک مشکل در پروتکل این برنامه عدم اطلاع سرور و کاربر شروع کننده استریم از وضعیت ارسال فایل و اطمینان از صحت انجام کامل آن است که برای این منظور میتوان دستوری به پروتکل اضافه کرد که در آن هر کاربر بعد از دریافت فایل به صورت P2P از کاربر دیگری، اتمام موفقیت آمیز این انتقال را به سرور اطلاع دهد، در صورتی که تعداد این پیام ها که توسط سرور دریافت شده برابر تعداد متقاضیان فایل بود یعنی این زنجیره به درستی کامل گشته است و در غیر این صورت سرور از اینکه کدام کاربران هنوز فایل را دریافت نکرده و یا اینکه زنجیره در کجا قطع شده مطلع خواهد شد.

2) در صورتی که چند کلاینت درخواست ارسال جریان داده‌ای را به سرور ارایه بدهند بهترین راهکار پیشنهادی شما چیست ؟

ساده ترین روش این است که اگر سرور در حال تشکیل زنجیره‌ی جریانی است در صورت دریافت درخواست استریم جدید در آن زمان به کاربر درخواست کننده یک پاسخ NACK با پیغام خطا سرور مشغول است برگرداند که روشی ساده اما غیرکارآمد است. روش دیگر این است که درخواست استریم جدید را بافر کرده و بعد از آزاد شدن سرور به ترتیبی که میتواند بر اساس زمان درخواست، الویت، حجم فایل یا موارد دیگر باشد به درخواست های موجود در بافر به ترتیب پاسخ دهد. روش دیگری که میتوان به کار برد تشکیل زنجیره‌های مختلف در عین واحد به صورت موازی است که در این روش در هر زنجیره بعد از ارسال یک چانک از فایل، کنترل سرور به زنجیره‌ی بعدی داده میشود و چانک زنجیره‌ی بعدی ارسال میشود که مدیریت این زنجیره های موازی و فایل های ارسال شده بزرگترین ایراد این روش و زمان کم پاسخگویی به هر درخواست مزیت این روش است.

3)پیشنهاد شما برای تشکیل بهینه یک زنجیره ارتباطی در این پروژه به صورتی که زمان ارسال کامل جریان به حداقل برسد، چیست ؟

میتوان برنامه را به گونه ای نوشت که هر کاربر به محض دریافت اولین چانک آن را به کاربر بعدی همزمان با دریافت دومین چانک ارسال کند که بدین صورت از ظرفیت آپلود و دانلود هر کلاینت در زمان حداکثر استفاده را کرده‌ایم. همچنین میتوان مقدار چانک ارسالی و دریافتی در این پروتکل را متناسب با میزان packet loss، حجم فایل و تعداد کاربران موجود در شبکه به صورت پویا انتخاب کرد. روش موثر بعدی که کمک زیادی به سرعت انتقال فایل در زنجیره میکند این است که سرور با دانستن

یا اندازه گیری پهنای باند کلاینت های موجود در این زنجیره، کاربران با پهنای باند بیشتر را در ابتدای زنجیره قرار دهد تا فایل به آنان زودتر برسد.

4)هنگام ارسال جریان داده‌ای احتمال قطع ارتباط کلاینتها وجود دارد پیشنهادی برای از دست نرفتن کامل زنجیره ارتباطی به وجود آمده طرح کنید ؟

برای حل این مشکل برنامه سمت سرور میتواند هر چند ثانیه یک بار به تمام کاربران موجود در شبکه دستور خاصی را برای اطلاع از متصل بودن آن کلاینت به شبکه داشته باشد، کلاینت در هنگام دریافت این دستور در جواب با ارسال ACK این دستور اعلام میکند که همچنان در شبکه متصل است. اما با تنظیم کردن timeout برای گوش دادن به جواب دستور سرور از سوی کلاینت اگر پاسخی دریافت نشد سرور متوجه ایجاد مشکل یا قطع ارتباط آن کلاینت شده و آن را از لیست کاربران خود پاک میکند و اگر در آن هنگام زنجیره‌ی داده‌ای در حال انتقال فایل است با اعلام خارج شدن کاربری از شبکه به کلاینت های موجود در آن زنجیره‌ی در حال انجام باعث بروزرسانی در زنجیره‌ی جاری خواهد شد.

5)با افزایش اندازه بسته‌های ارسالی در زنجیره ارتباطی، زمان دریافت بسته‌ها را در آخرین کلاینت بررسی نمایید.

از آنجایی که این انتقال تحت بستر پروتکل UDP انجام میشود که در این پروتکل از دست رفتن بسته‌ای تاثیری بر سرعت دریافت کل فایل نخواهد داشت، با بزرگتر کردن اندازه بسته ها به دلیل کاهش سربار ناشی از Header، طبق فرمول delay=L/R باعث کاهش میزان زمان دریافت فایل به صورت کامل میشود.
اما اگر انتقال فایل ها تحت بستر پروتکل TCP انجام میشد بسته به میزان ترافیک و packet loss مسیر تا جایی افزایش مقدار هر بسته به کاهش زمان دریافت فایل منجر میشد و بعد از آن مقدار با افزایش میزان هر بسته با افزایش حتی نمایی تاخیر ارسال فایل مواجه خواهیم شد.

پیاده سازی برنامه سمت کلاینت

برنامه سمت کلاینت شامل کد اصلی برنامه که همواره در حال اجرا و آماده‌ی دریافت دستورات از سمت کاربر و ارسال آن به سمت سرور در قالب یک پروتکل TCP است میباشد، همچنین این برنامه شامل یک تابع receive است که به صورت Thread فراخوانی میشود و دستوراتی که از سمت سرور یا داده اصلی که از سوی کاربران ارسال میشود را دریافت میکند.
در کد اصلی برنامه ابتدا کاربر باید با انتخاب نام مناسب و غیر تکراری اقدام به ثبت نام در سرور کند و بعد از آن وارد حلقه نامتناهی بعدی شده که تنها در آن زمان قادر است دستورات کنترلی خود را ارسال کند. در تابع receive نیز یک حلقه‌ی نامتناهی وجود دارد که در ابتدای آن به سوکت TCP برقرار شده با سرور گوش میدهد و متناسب با دستور دریافت شده کارهای زیر را انجام خواهد داد:

1)دریافت تایید خروج از شبکه (Bye#OK):

بعد از موافقت سرور با این دستور کلاینت برای خروج از این شبکه سوکت خود با سرور را بسته و دستور()sys.exit را برای خروج از برنامه سمت کلاینت اجرا میکند.

2)موافقت با شروع استریم داده (StreamReq#OK#host#IP{next chain}):

در این حالت کلاینت با ایجاد یک سوکت UDP اقدام به ارسال فایل به اولین کلاینت موجود در این زنجیره(که آدرس پورت آن را نیز در ادامه همین ACK دریافت کرده) خواهد کرد.

3) قبول یا رد دریافت یک فایل و حضور در زنجیره آن فایل (StreamReq#[filename]):

اگر کاربر مایل به دریافت فایل ذکر شده در درخواست باشد ابتدا یک سوکت UDP ایجاد کرده و سپس شماره پورت آن را همراه با ACK درخواست فایل در قالب زیر به سرور، تحت همان اتصالTCP میفرستد:

StreamReq#OK#HostName#UDP_port

سپس بر روی سوکت UDP ایجاد شده منتظر دریافت فایل مورد نظر میشود.

بعد از دریافت کامل فایل و دریافت درخواست دیگری از سرور مبنی بر ارسال فایل دریافت شده به کاربر بعدی این کلاینت فایل دریافت شده را به کاربر بعدی با شماره پورت UDP که سرور به آن اعلام کرده میکند(مگر اینکه این کلاینت آخرین کلاینت موجود در زنجیره باشد).

پیاده سازی برنامه سمت سرور

در برنامه سمت سرور در کد اصلی برنامه یک حلقه نامتناهی وجود دارد که درخواست های اتصال TCP را از سوی کاربران قبول کرده و برای هر کدام یک Thread ایجاد کرده و سوکت برقرار شده با آن کاربر را به تابع اصلی که به ازای هر کاربر یک Thread از آن ایجاد میشود را ارسال میکند، در این تابع بعد از ثبت نام کاربر توسط تابع Register، تابع اصلی سوکت این اتصال را وارد لیست سراسری ثبت نام شدگان در این شبکه کرده و وارد حلقه نامتنهای بعدی و اصلی برنامه میشود، در حلقه این تابع نیز بسته به دستور کنترلی دریافت شده وارد قسمت شرطی مربوطه میشود:

1)درخواست به اشترام گذاری فایل (StreamReq#[filename]):

سرور بعد از دریافت این دستور از تمام کلاینت های موجود در شبکه (به غیر از درخواست کننده اصلی) میپرسد که آیا این فایل را میخواهید یا نه؟

2) اعلام موافقت یا عدم موافقت کاربر برای دریافت فایل مورد نظر(StreamReq#OK/NOK#[UDP]):

در صورت موافقت هر کلاینت با فرستادن دستور StreamReq#OK#HostName#UDP_port آمادگی خود برای دریافت فایل را اعلام میکند و سرور شماره سوکت UDP آن کاربر را در لیست زنجیره‌ی داده قرار میدهد بعد از اتمام نظرسنجی توسط

سرور حال کلاینت اقدام به تشکیل زنجیره داده‌ی مورد نظر کرده و به هر کلاینت موجود در این زنجیره شماره پورت UDP کلاینت بعدی را برای ارسال P2P اعلام میکند.

3)درخواست خروج از شبکه (Bye):

در صورت نبودن مشکلی سرور سوکت درخواست کننده‌ی این دستور را از لیست ثبت نام شدگان در این شبکه خارج کرده و با ارسال دستور Bye#OK این موضوع را به درخواست کننده اعلام میکند.