دانشکده‌ی مهندسی کامپیوتر

شبکه‌های کامپیوتری مدرس: مهدی جعفری

تمرین عملی لایه Network

گردآورندگان: مهراد میلانلو - امیرمهدی نامجو جمعه ۶ بهمن ۱۴۰۲

در این سوال می‌توانید از هر زبان برنامه نویسی استفاده کنید.

۱ UDP Hole Punching

فرض کنید که یک سرویس تماس تصویری و صوتی آنلاین مانند Google Meet یا Skyroom دارید. اولین روشی که برای برقراری ارتباط بین دو فرد به ذهن می‌رسد این است که ترافیک موردنظر به کمک یک سرور به هر فرد فرستاده شود یا به اصطلاح relay یا proxy شود. این کار مسلما توان مصرفی زیادی را درگیر می‌کند و زمانی که تعداد کاربرها بالا رود، باید سرورهای قوی‌تری برای پراکسی کردن داده‌ها قرار بدهیم. اما چه می‌شد اگر می‌توانستیم بدون یک سرور میانی، دو شخصی که می‌خواهند با هم حرف بزنند را به هم متصل کنیم؟

امروزه بسیاری از سیستم‌عامل‌ها به یک Firewall مجهز هستند. این فایروال‌ها به‌صورت پیش‌فرض اجازه نمی‌دهند که هیچ کانکشنی از خارج شبکه‌ی شما به کامپیوتر زده شود. از طرف دیگر، بسیاری از شبکه‌های خانگی پشت Network Address Translation (NAT) هستند. NAT ها این اجازه را به کامپیوترهای متصل به آن می‌دهند که همگی آن‌ها یک IP مشترک داشته باشند.

۱.۱ چرا NAT؟

در دنیای امروزی اینترنت هر شخصی که می‌خواهد با دیگران ارتباط برقرار کند باید یک آدرس IP داشته باشد. استاندارد IPv4 تنها حدود ۴ میلیارد IP موجود دارد و در حال حاضر بیش‌تر از ۲ میلیارد وسیله‌ی متصل به اینترنت وجود دارد؛ یعنی به‌وضوح با کمبود تعداد آدرس مواجه هستیم. برای همین روش NAT معرفی شد که به کمک آن می‌توان به چندین کامپیوتر یک IP را اختصاص داد. همان‌طور که می‌دانید، این روش بدین صورت عمل می‌کند که شما باید به یک Router (مسیریاب) یک IP بدهید که در اینترنت بتواند با بقیه حرف بزند. به این آدرس، آدرس Public IP می‌گویند. همچنین به هر یک از کامپیوترهایی که به شبکه‌ی داخلی شبکه‌ی شما وصل هستند یک آدرس Private IP اختصاص می‌دهد. این IP ها به نحوی یک IP مجازی هستند که در دنیای اینترنت به دستگاه خاصی داده نشده‌اند و همیشه برای استفاده‌ی آن‌ها NAT ها رزرو شده‌اند.

اما هم‌چنان یک مشکل وجود دارد. فرض کنید که دو کامپیوتر که به NAT وصل هستند، همزمان از یک پورت بخواهند به یک سایت واحد متصل شوند. در این‌جا نمی‌توان تنها IP ها را تغییر داد چرا که درخواست دریافتی از سمت کامپیوتر مقصد معلوم نیست که برای کدام یک از کامپیوترهای Local Network هستند. در این‌جا مجبور هستیم که Port را نیز عوض کنیم.

به‌صورت کلی بسیاری از NAT ها دقیقا به همین شکل کار می‌کنند. فرض کنید که در ابتدا کامپیوتری با آدرس Private IP 10.1.1.5 می‌خواهد از پورت 12345 یک بسته به پورت 54321 آدرس IP 2.3.4.5 بفرستد. زمانی که این بسته را سیستم‌عامل می‌سازد، آدرس مبدا را همان 10.1.1.5 قرار می‌دهد. زمانی که NAT این بسته را دریافت می‌کند، آدرس 10.1.1.5 و پورت 12345 را به آدرس Public IP و یک پورت رندوم تغییر می‌دهد و سپس بسته به اینترنت فرستاده می‌شود. همزمان NAT در حافظه‌ی خودش ذخیره می‌کند که اگر یک بسته از 54321 : 2.3.4.5 آمد، آن را برای 12345 : 10.1.1.5 بفرستد.

در این روش، نکته این است که تنها بسته‌های دریافتی از 54321 : 2.3.4.5 به کامپیوتر مذکور منتقل می‌شوند. پس امکان ندارد که هرکسی در اینترنت بتواند سرخود برای کامپیوترهایی که در پشت NAT هستند بسته بفرستد. حتما باید در ابتدا کامپیوتری که پشت NAT است با کامپیوتری که می‌خواهد ارتباط برقرار کند اول یک بسته فرستاده باشد که NAT بداند بسته‌های ورودی را به کجا ارسال کند. به همین دلیل به‌نظر می‌رسد برقراری ارتباط مستقیم بین دو کامپیوتر که پشت NAT هستند امکان‌پذیر نباشد...

۲.۱ ظهور ایده!

فرض کنید شما یک بسته به 1000 : 1.2.3.4 می‌فرستید. در این حالت NAT به‌خاطر می‌سپارد که هر بسته‌ای از 1000 : 1.2.3.4 باید به کامپیوتر شما فرستاده‌شود. حال فرض کنید دقیقا از همان IP و پورت کامپیوتر خودتان یک بسته به 5.6.7.8 می‌فرستید. حال NAT، همان IP:Port قبلی خودتان را به 2000 : 5.6.7.8 مپ می‌کند. پس اگر جوری دو کامپیوتری که پشت NAT هستند بتوانند برای IP:Port هم‌دیگر یک پیام بفرستند، NAT هایشان طوری تنظیم می‌شود که می‌توانند با هم به‌صورت مستقیم و بدون واسطه حرف بزنند!

به کمک همین موضوع ایده‌ی UDP Hole Punching مطرح شد. دلیل وجود پروتکل UDP این است که این روش به‌خاطر stream base بودن TCP، به‌طور معمولی جواب نمی‌دهد. در این پروتکل یک سرور دیگر نیز وجود دارد که باید از سمت هر دو peer که می‌خواهند به هم متصل شوند قابل دسترسی باشند. این سرور را STUN می‌نمایم. UDP Hole Punching بدین صورت عمل می‌کند که در ابتدا هر دو peer به STUN می‌گویند که می‌خواهیم با هم ارتباط برقرار کنیم. با این کار NAT هر دو peer به صورتی تنظیم می‌شود که بسته‌ها فقط بتوانند به سرور STUN برسند و دریافت شوند. سپس STUN به هرکدام از peer ها آدرس peer دیگر را می‌دهد و هر یک از peer ها به آن آدرس یک بسته می‌فرستند. با این کار NAT هرکدام طوری تنظیم می‌شود که بتوانند برای هم‌دیگر پیام بفرستند و در نتیجه سرور STUN دیگر کارایی ندارد؛ زیرا اکنون هر دو peer می‌توانند بدون واسطه با هم حرف بزنند.

۳.۱ خیر بدون شر؟

هیچ خیری بدون شر نمی‌آید! UDP Hole Punching دو مشکل بزرگ دارد:

۱. در بعضی از انواع NAT ها، به اسم Symmetric NAT، روش UDP Hole Punching کار نمی‌کند. در این نوع NAT ها با عوض شدن آدرس مقصد پورتی که NAT به‌عنوان پورت مبدا بازنویسی می‌کند نیز عوض می‌شود و برای همین آدرسی که دست STUN هست با آدرسی که به peer دیگر فرستاده می‌شود فرق دارد. این نوع NAT معمولا در اپراتورهای تلفن همراه دیده می‌شود.

۲. لو رفتن IP هر یک از peer ها! فرض کنید شما می‌خواهید آدرس IP یک شخص را پیدا کنید. برای این کار یکی از کارهایی که می‌توانید بکنید این است که به کمک یک برنامه‌ی مکالمه‌ی آنلاین به آن شخص زنگ بزنید و با این کار UDP Hole Punching اتفاق می‌افتد و IP شخص موردنظر لو می‌رود.

۲ خواسته های تمرین

برای یادگیری UDP Hole Punching، در این تمرین می‌خواهیم یک سامانه سرور_کلاینت که از این روش استفاده کنند پیاده‌سازی کنیم.

برای این منظور، د شما باید یک STUN Server ساده پیاده‌سازی کنید که وظیفه دارد با دریافت درخواست از کلاینت‌ها، ابتدا آی‌پی پابلیک آن کلاینت را به همان کلاینت اعلام کرده و سپس بعد از دریافت درخواست از کلاینت بعدی، با انتقال دادن آی‌پی این دو کلاینت به یکدیگر، شرایط را برای برقراری ارتباط P2P مستقیم بین این دو فراهم کند. توجه کنید که سرور شما باید مدام به کار خود ادامه دهد و بعد از ایجاد ارتباط بین این دو کلاینت، همچنان منتظر درخواست‌های بعدی از کلاینت‌های دیگر بماند و آن‌ها را هم به همین ترتیب به هم متصل نماید.

شیوه کارکرد کلاینت هم به این صورت خواهد بود که در ابتدای راه‌اندازی یک رشته از کاربر دریافت می‌کند. سپس با STUN Server (که آی‌پی و پورت آن مشخص است) ارتباط برقرار کرده و بعد از دریافت آی‌پی خودش و آی‌پی کلاینت دیگری که باید با آن ارتباط برقرار کند، اقدام به ارسال پیام مستقیم به کلاینت دیگر می‌کند. بدین ترتیب باید انتقال پیام از طریق تکنیک UDP Hole Punching با موفقیت بین این دو صورت بگیرد. پیام منتقل شده از هر کلاینت به کلاینت دیگر، شامل رشته‌ای که در ابتدا دریافت می‌شود و تاریخ و زمان ارسال پیام است.

لازم به ذکر است که لازم نیست تمامی جزییات پروتکل STUN به طور ریز و دقیق در سرور پیاده‌سازی شده رعایت بشود، بلکه صرفا عملکرد اصلی که ایجاد ارتباط بین کلاینت‌ها است باید به درستی پیاده‌سازی بشود.

توجه کنید که با توجه به این که شما باید یک STUN Serverپیاده‌سازی کنید، انتظار نمی‌رود که برنامه شما برای سیستم‌هایی که پشت Symmetric NAT قرار دارند کار کند و نیازی به رفع مشکل در این حالت نیست.

امتیازی: برای دریافت نمره امتیازی تمرین، سیستم سرور خود را طوری پیاده‌سازی کنید که به جای این که هر کلاینت را به کلاینت بعدی خود متصل کند، در ابتدا یک ID از کلاینت‌ها تحویل بگیرد و سپس بعدا بسته به درخواست کلاینت، آن را به کلاینتی با ID مشخص شده متصل کند. پیاده‌سازی درست بخش امتیازی تا سقف ۲۰ درصد نمره تمرین مشمول نمره امتیازی می‌شود.

۳ تحویل دادنی‌ها

فرمت فایل هایی که تحویل می‌دهید باید به صورت زیر باشد:

۱. فایل README که نام، نام‌خانوادگی، شماره دانشجویی و نحوه اجرا کردن کدهای شما در آن ذکر شده باشد.

۲. کدهای خود را می‌توانید به هر زبانی می‌خواهید بنویسید ولی پیشنهاد ما پایتون است.

۳. یک فایل Makefile که پروژه‌ی شما رو بیلد می‌کند. در صورتی که از dependecy خاصی استفاده می‌کنید حتما آن را در Makefile خود نصب کنید. به طور کلی باید امکان اجرای کد شما براساس توضیحات و فایل‌هایی که قرار داده‌ای به راحتی امکان پذیر باشد.

۴. یک فایل PDF شامل گزارشی از نحوه عملکرد کد شما، درک شما از چگونگی کارکرد UDP Hole Punching و تصاویری از اجرای درست برنامه روی سیستم (های) شما. نیازی به توضیحات مفصل و طولانی زیاد در این فایل نیست، ولی باید محتوای فایل طوری باشد که مشخص بشود شما مفهوم اصلی این روش را درک کرده‌اید و با خواندن آن بتوان متوجه شد چرا کد شما به درستی این روش را پیاده‌سازی کرده است.

۴ تست و ارزیابی

در صورتی که بخواهید تمرین را به صورت واقعی تست کنید، نیاز است که سرور خود را روی یک سرور یا سیستم واقعی که آی‌پی آن به صورت عمومی در دسترس باشد اجرا کرده و سپس با استفاده از دو کامپیوتر به عنوان کلاینت که به اینترنت‌های ثابت خانگی یا اینترنت دانشگاه متصل هستند (پشت NAT هستند)، اقدام به ارسال پیام کنید. با این وجود با توجه به این که ممکن است این کار برای همه امکان پذیر نباشد، اگر صرفا این موضوع را روی سیستم خود و با IPهای داخلی آن تست کنید و فرآیند به درستی انجام بشود، کافیست. ضمن این که می‌توانید از طریق ابزارهای ماشین‌مجازی نظیر Virtual Box یا VMWare، سیستم اتصالی کامپیوترها پشت NAT را شبیه‌سازی کنید و در صورتی که می‌خواهید آن را روی ماشین مجازی‌های خود تست کنید.

موفق باشید