# مقدمه

در این تمرین قصد داریم تا شما را با پیاده‌سازی یک نرم‌افزار پروکسی آشنا کنیم. امروزه استفاده از پروکسی‌ها در لایه‌های مختلف اجتناب ناپذیر شده است. به‌عنوان مثال همان‌طور که در درس خواندید، پروتکل UDP فاقد مکانیسم‌های کنترل ازدحام است و می‌تواند باعث شود تا ارسال کنندگان داده با ارسال زیاد بسته‌های UDP باعث کاهش کارایی سیستم‌هایی شوند که از پروتکل TCP به عنوان پروتکل لایه انتقال خود استفاده می‌کنند. به همین علت ممکن است در برخی از شبکه‌ها استفاده از پروتکل UDP محدود شود. به این منظور کاربران این شبکه‌ها از پروکسی‌های لایه حمل استفاده می‌کنند که بسته‌های UDP را به بسته‌های TCP تبدیل کرده و پس از دریافت جواب این تبدیل معکوس را مجددا انجام می‌دهد.

## قوانین

۱. پیاده‌سازی تنها با زبان جاوا می‌تواند صورت می‌گیرد.

۲. ارسال و دریافت بسته‌ها تماما با استفاده از **Socket**ها در جاوا صورت می‌گیرد و قسمت‌های مربوط به ساخت، گوش دادن، ارسال و دریافت می‌بایست در کد شما بوده و قابل رویت باشند.

۳. بسته‌های HTTP می‌بایست به وسیله‌ی شما و بدون استفاده از هیچ کتابخانه‌ای ساخته و پردازش شوند. قسمت‌های مربوط می‌بایست در کد شما به صورت مشخص قابل رویت باشند.

۴. دقت داشته باشید در موارد بالا استفاده از کتابخانه‌های استاندارد جاوا نیز مجاز نمی‌باشد.

# شرح تمرین

این تمرین از دو بخش تشکیل شده است: یک بخش برنامه *پروکسی* و بخش دیگر ابزاری است که به عنوان *سرویس گیرنده* به پروکسی متصل می‌شود. برای بخش *سرویس دهنده* از سرورهای واقعی موجود در اینترنت استفاده می‌شود و نیازی به پیاده سازی نیست. برنامه پروکسی بین سرویس گیرنده و سرویس دهنده قرار دارد و درخواست‌های سرویس گیرنده را به قالب مناسب تبدیل کرده و برای سرویس دهنده ارسال می‌کند.

## پروکسی

برنامه پروکسی باید قادر باشد، پیغام‌های UDP ارسال شده از طرف سرویس گیرنده را بر روی اتصال TCP به سمت سرویس دهنده که بر روی پورت مشخصی در حال گوش کردن است ارسال کند. جواب درخواست‌ها بر روی اتصال UDP به سرویس گیرنده ارسال می‌شود..

ابزار پروکسی باید حداقل از گزینه‌های زیر برخوردار باشد:

proxy –s sourceHost:sourcePort

یک مثال از دستور فوق می‌تواند مشابه شکل زیر باشد:

proxy –s 127.0.0.1:80

در صورت اجرای این دستور فوق برنامه شما کارهای زیر را انجام می‌دهد:

* تبدیل بسته‌های UDP به جریان TCP
* دریافت بسته‌های UDP در آدرس 127.0.0.1:80 (به این معنی که برنامه‌ی شما بر روی پورت ۸۰ و آدرس IP 127.0.0.1 عمل binding را انجام می‌دهد)
* دریافت بسته‌های UDP از سرویس گیرنده، خارج کردن محتوای درخواست و ارسال آن در قالب بسته‌های TCP به سرویس دهنده
* دریافت بسته‌های TCP از سرویس دهنده، خارج کردن محتوای آن، ایجاد بسته یا بسته‌های UDP و ارسال آن به سرویس گیرنده

## سرویس گیرنده

همان‌گونه که مشخص است برنامه‌های نوشته شده فعلی قادر به کارکردن با پروکسی فوق نیستند. برای همین ابزاری مناسب این پروکسی باید ایجاد شود. این ابزار از دو کاربرد اینترنت پشتیبانی می‌کند:

* کاربرد HTTP بر روی پروتکل UDP

### کاربرد HTTP بر روی پروتکل UDP

برنامه سرویس‌گیرنده شما مشابه یک مرورگر اینترنت، البته با قابلیت‌های بسیار کمتر عمل می‌کند. برنامه شما باید به عنوان ورودی یک درخواست **HTTP استاندارد** (این درخواست را می‌بایست بدون استفاده از کتابخانه‌ها تولید نمایید.) را از کاربر دریافت کند و آن را در قالب بسته UDP به پروکسی ارسال کند. برای سادگی صرفا برنامه شما باید قادر به پشتیبانی از GET می‌باشد. مثالی از ورودی به برنامه در زیر آمده است:

GET / HTTP/1.1

Host: aut.ac.ir

در این حالت برنامه شما بسته UDP را ایجاد کرده و آن را به پروکسی که بر روی پورت UDP مشخصی تنظیم شده است ارسال می‌کند. پروکسی مطابق با درخواست دریافت شده، یک ارتباط TCP با میزبان با نام دامنه aut.ac.ir بر روی پورت پیش‌فرض 80 برقرار می‌کند. سپس پروکسی درخواست HTTP را به وب سرور مورد نظر در پورت 80 ارسال می‌کند و منتظر جواب می‌ماند. پس از دریافت کامل جواب، بسته یا بسته‌های UDP را ایجاد کرده و جواب را با استفاده از این بسته‌ها به سمت سرویس گیرنده ارسال می‌کند. برنامه شما پس از دریافت کامل بسته‌ها، یک فایل HTML ایجاد کرده و محتویات بسته‌ها را در آن قرار می‌دهد. این فایل باید به صورت کامل قابل خواندن توسط مرورگر باشد و بدون خطا نمایش داده شود.

* برنامه شما باید قابلیت پشتیبانی از کدهای 301، 302 و 404 را داشته باشد و با نمایش دادن یک پیغام کاربر را از نتیجه درخواست آگاه کند. در صورتی که جواب کد 301 یا 302 را داشت، درخواست برای Location جدید باید به صورت خودکار در برنامه شما آن‌قدر تکرار شود تا به کد 200 برسید.
* لزومی به دریافت اشیا دیگر به جز شی اصلی ندارید.
* امتیاز اضافه: تقاضاهایی که یکبار قبلا انجام شده است را در پروکسی Cache کرده و از تکرار آن جلوگیری کنید. انتخاب معماری و نحوه عملکرد کش بر عهده خودتان است.
* به جز موارد الزامی گفته شده، نحوه انتخاب سایر فیلدهای درخواست‌ها و پروتکل‌ها بر عهده خودتان است.
* پروکسی مقصد ارتباط را از محتوای بسته دریافتی تشخیص می‌دهد و از قبل مشخص نمی‌شود.

همانطور که می‌دانید اندازه‌ی بسته‌های UDP محدودیت داشته و شما نمی‌توانید هر مقدار داده‌ای را از طریق آن‌ها ارسال نمایید. پیام‌های TCP که پروکسی دریافت می‌کند، ممکن است اندازه‌ی بزرگ داشته باشند و نتوان آن‌ها را به وسیله‌ی یک پیام UDP ارسال نمود. بنابراین نیاز است که داده‌ی دریافتی از طریق TCP در پروکسی به قطعات مختلف شکسته شده و این قطعات از طریق UDP به سرویس‌گیرنده ارسال شوند. به این عمل در این پروژه segmentation خواهیم گفت.

ارتباط TCP بین سرویس‌دهنده و پراکسی قابل اطمینان می‌باشد، بنابراین برای اینکه تجربه کاربر از ارتباط راضی‌کننده باشد، می‌بایست ارتباط بین پراکسی و سرویس‌گیرنده که مبتنی بر UDP می‌باشد را نیز قابل اطمینان سازیم. برای این امر نیاز است که روش برای تشخیص خطا مانند Parity و یک روش ARQ در ارتباط میان پراکسی و سرویس‌گیرنده پیاده‌سازی شود. انتخاب این روش‌ها بر عهده شما خواهد بود.

موفق باشید