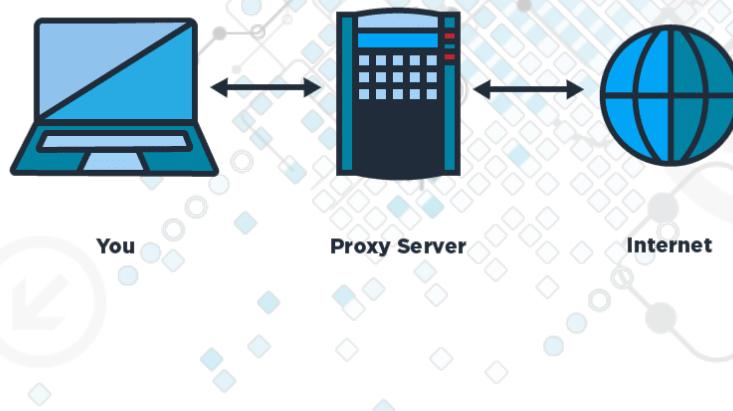


## پروکسی سرور



### چکیده

شما در این پروژه باید یک proxy server پیاده‌سازی کنید و با تنظیم پروکسی مرورگر بر روی ip و port برنامه‌ی خود، مرورگر بجای ارسال مستقیم بسته‌ها به server مقصد، تمامی بسته‌ها را برای برنامه‌ی شما ارسال می‌کند. برنامه‌ی شما با پارس کردن بسته و عمل تغییرات خاصی، آن را برای سرور مد نظر کاربر ارسال می‌کند و بعد از گرفتن پاسخ از سرور اصلی، پاسخ را برای مرورگر ارسال می‌کند. همچنین در ادامه، قابلیت‌هایی مانند caching، ارسال ایمیل اخطار به ادمین، تزریق کد و حفظ حریم شخصی به پروژه اضافه خواهند شد.

### Proxy server ها:

پروتکل HTTP یک پروتکل لایه کاربرد شبکه مبتنی بر پروتکل TCP است. همان‌طور که از نامش بر می‌آید برای ارسال و دریافت ابرمن استفاده می‌شود. اولین و اصلی‌ترین مورد استفاده‌ی این پروتکل، در وب جهانی است.

به طور معمول client (معمولًا مرورگر کاربر) به صورت مستقیم با سرور ارتباط برقرار می‌کند و اطلاعات را دریافت و ارسال می‌کند؛ ولی در شرایط خاصی، کلاینت از طریق یک موجودیت سوم که پروکسی نام دارد با سرور مقصد ارتباط برقرار کرده و به این شکل ارتباط با سرور به صورت غیرمستقیم برقرار خواهد شد.

در ساده‌ترین حالت، پروکسی فقط بسته‌ها را از مرورگر دریافت می‌کند و به سمت سرور اصلی ارسال می‌کند و بعد از گرفتن جواب، آن‌ها را برای مرورگر ارسال می‌کند؛ اما می‌تواند بجز این، کاربردهای دیگری هم داشته باشد که در ادامه به برخی از آن‌ها اشاره می‌کنیم.

### Caching

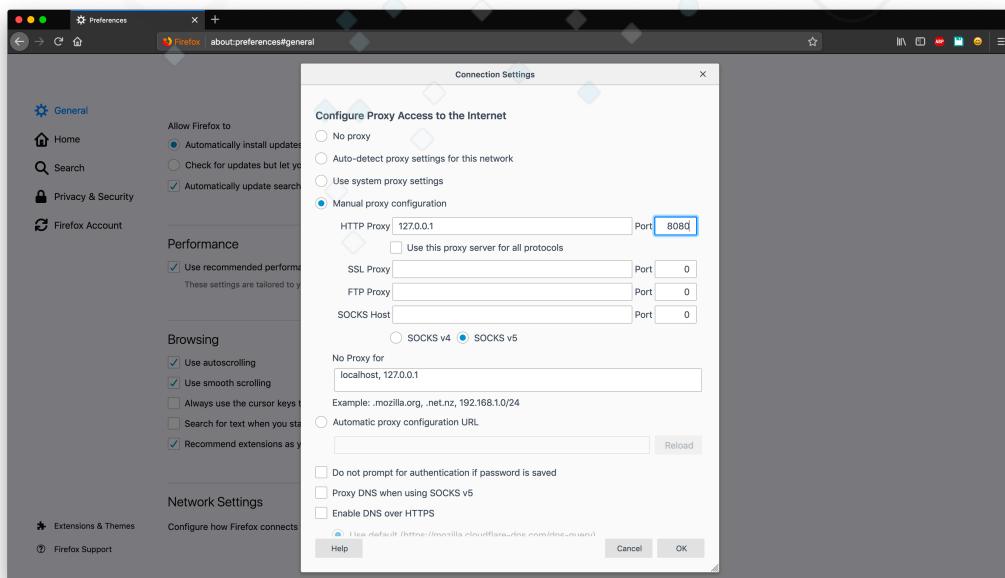
پروکسی می‌تواند با ذخیره کردن بعضی از داده‌ها که کمتر تغییر می‌کنند یا حجم بیشتری دارند، مانند عکس‌ها، باعث افزایش سرعت بارگذاری و کاهش تاخیر در دریافت داده‌ها شود. مثلاً مدیر شبکه دانشگاه می‌تواند با راهاندازی یک پروکسی در شبکه داخلی، باعث شود بعضی از منابع اینترنت که اکثر کاربران وب آنها را دریافت می‌کنند به صورت cache ذخیره شود؛ در این صورت حجم زیادی از ترافیک‌های تکراری حذف می‌شود و از طرفی چون منابع در سرورهای دانشگاه ذخیره شده‌اند باعث افزایش سرعت استفاده‌ی کاربران می‌شود.

## Privacy

شما به طور معمول اطلاعات زیادی را برای وبسایت‌ها ارسال می‌کنید که این اطلاعات می‌تواند هم برای صاحب وبسایت‌ها و هم افرادی که در شبکه قرار گرفته‌اند مفید باشد و به آن‌ها کمک کند تا از هویت شما آگاه شوند. در ساده‌ترین حالت، شما در زمان وب‌گردی در header ارسالی خود اطلاعات مربوط به سیستم‌عامل و مرورگر خود (User agent) را برای سرور ارسال می‌کنید. وبسایت‌ها می‌توانند با سواسفه‌های HTTP از این اطلاعات کاربران را ردیابی کنند. کاربران می‌توانند با استفاده از پروکسی این مشکل را رفع کنند؛ چون با این کار درخواست‌ها به پروکسی ارسال می‌شود و پروکسی می‌تواند با تغییر این فیلد از نشت اطلاعات شخصی کاربران جلوگیری کند.

## تعريف پروژه

شما در این پروژه یک web proxy ساده پیاده‌سازی می‌کنید. پروکسی شما روی پورتی که از طریق فایل config در زمان اجرا به برنامه داده می‌شود یک TCP Socket باز می‌کند و روی پورت مورد نظر منتظر می‌ماند. سپس با تغییر تنظیمات proxy server مرورگر و تنظیم ip و port آن مطابق با اطلاعات پروکسی خود، مرورگر با این واسطه ارتباط برقرار می‌کند و درخواست خود را برای شما ارسال می‌کند. برنامه شما بعد از گرفتن درخواست از مرورگر باید درخواست را parse کند و بعد از ایجاد درخواست جدید و مشخص کردن مقصد اصلی، درخواست را برای آن ارسال و بعد از گرفتن پاسخ، آن را برای مرورگر شما ارسال کند.



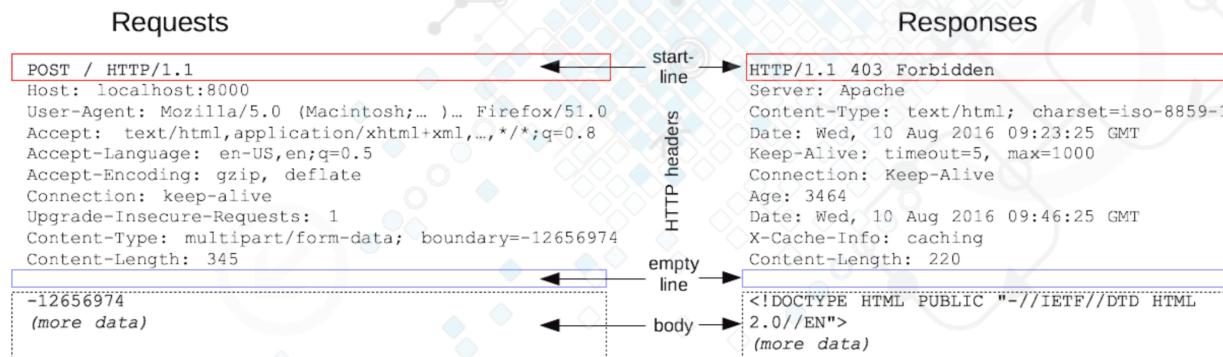
## تنظیمات مربوط به پروکسی در مرورگر فایرفاکس

برای سادگی کار می‌توانید همواره از پروتکل HTTP/1.0 web server proxy و استفاده کنید. پروتکل HTTP/1.1 از ارتباط پایا استفاده می‌کند که ممکن است برایتان مشکلاتی به وجود بیاورد. در این صورت در زمان دریافت درخواست از نوع HTTP/1.1 کافیست Header آن را به HTTP/1.0 تبدیل کنید و برای web server ارسال کنید. البته proxy server شما باید بتواند از هر دو پروتکل HTTP/1.0 و HTTP/1.1 خودش و مرورگر پشتیبانی کند.

## پیاده‌سازی

قبل از شروع کار باید بدانید زمانی که نام وب‌سایتی را در URL-bar مورگر خود وارد می‌کنید و دکمه enter را می‌فشارید چه اتفاقی می‌افتد:

همان‌طور که در درس آموختید، در حالت ساده مورگرها و web serverها برای ایجاد ارتباط و رد و بدل کردن اطلاعات مورد نیاز، از پروتکل به نام HTTP استفاده می‌کنند. HTTP یک پروتکل لایه کاربرد بر روی پروتکل TCP است. (یعنی صحت اطلاعات ارسالی و دریافتی تضمین شده است). برای مثال، برای گرفتن صفحه اصلی یک وب‌سایت (index.html) بعد از وارد کردن نام آن و زدن دکمه enter، مورگر یک socket از نوع TCP با سرور آن وب‌سایت روی یک پورت مشخص (به صورت پیش‌فرض پورت ۸۰ برای ارتباطات از نوع HTTP استفاده می‌شود) برقرار می‌کند. بعد از ایجاد تونل TCP، مورگر باید درخواست خود که از پروتکل HTTP پیروی می‌کند را ارسال کند. پروتکل HTTP ساختاری مانند زیر دارد:



دق کنید که در بسته‌های HTTP برای رفتن به خط بعد از ترکیب دو کاراکتر \r\n استفاده شده است.

در خط اول، اولین بخش method درخواست را مشخص می‌کند که اغلب متدهای get, post, patch, delete، امروزه استفاده می‌شوند و برای هر کدام عملکرد متفاوتی تعریف می‌شود. در قسمت دوم path دقیق درخواست می‌آید که در این مثال فایل index.html مورد درخواست بوده است؛ و بعد از آن شماره نسخه درخواست درج می‌شود و در انتهای خط \r\n می‌آید که به خط بعد می‌رویم.

در خطوط بعدی هر header در یک خط آمده است که با علامت : از مقدار آن جدا شده است و در انتهای هر خط \r/n می‌آید. در بعضی مواقع، مانند متد post، می‌توانیم در بسته‌ی ارسالی data هم داشته باشیم که در انتهای بسته و با یک خط خالی (\r/n) از خطوط header جدا می‌شود.

برای اطلاعات دقیق‌تر بخش ۲-۲ کتاب مرجع درس را مطالعه کنید. (صفحه ۹۸)

زمانی که تنظیمات proxy server روی مورگر شما تنظیم شده باشد، بسته‌ی HTTP با کمی تغییرات به سرور ارسال می‌شود و بعد از آن بسته‌ی اصلی برای وب‌سرور ارسال می‌شود و پروکسی پاسخ را برای مورگر ارسال می‌کند. برای اینکه بفهمید مورگر شما چه بسته‌ای را برای پروکسی تنظیم شده ارسال می‌کند می‌توانید از ابزار netcat استفاده کنید. Netcat ابزاری ساده برای ارسال و گرفتن بسته‌های UDP و TCP است. با اجرای netcat به شکل رویرو nc روی پورت داده شده منتظر می‌ماند و در صورتی که بسته‌ای برای آن ارسال شود آنرا در terminal چاپ می‌کند.

```
nc -l -p <port-number> -v
```

حال با تنظیم پروکسی در مورگر خود روی ip و port سیستم خود و پورت تنظیم شده و درخواست یک وب‌سایت که از پروتکل HTTP استفاده می‌کند می‌توانید بسته‌ی ارسال شده توسط مورگر را در netcat مشاهده کنید.

مشاهده می کنید که بسته های ارسالی به proxy server تفاوت کوچکی با بسته های معمولی دارند. (با مثال قبلی از پروتکل HTTP مقایسه کنید)

```
GET http://www.amazon.com/index.html HTTP/1.1  
Host: www.amazon.com  
Proxy-Connection: Close  
Connection: Close
```

زمانی که از proxy استفاده نمی کنید تنها قسمت های بعد از path و بدون host name آورده می شود؛ در غیر این صورت، آدرس وب سایت به صورت کامل درج می شود.

همچنین یک سرآیند proxy-connection هم به بسته اضافه می شود که همانند سرآیند connection عمل می کند با این تفاوت که وضعیت ارتباط بین مرورگر و proxy server را مشخص می کند. (نیاز به پیاده سازی عمل کرد این قسمت ندارید و همواره مقدار آن را close فرض کنید).

برنامه ای شما تنها باید قسمت host name proxy-connection را از آن حذف کند و برای web server آدرس و سرآیند proxy-connection را از آن حذف کند و برای ارسال کند و جواب را بدون تغییر به مرورگر برگرداند.

**نکته:** برای پیاده سازی می توانید از هر زبان دلخواهی استفاده کنید با این شرط که آن زبان قابلیت استفاده کردن از socket TCP را داشته باشد ولی پیشنهاد ما زبان پایتون و بعد از آن جاوا است. (استفاده از مازل های سطح بالا برای ارتباطات مجاز نمی باشد، مثلاً نمی توانید از مازل request در پایتون که بسته های http را می سازد استفاده کنید)

برای پیاده سازی پروژه قبل از هر کار باید فایل config با فرمت json را که کنار فایل اجرایی شما قرار داده شده است را بخوانید و آن را parse کنید، سپس باید روی پورتی که در فایل config آمده است یک socket TCP ایجاد کنید تا منتظر اتصال مرورگر بماند؛ بعد از آن باید تنظیمات proxy server مرورگر در حالت HTTP را روی ip و port برنامه خود تنظیم کنید؛ برای این کار می توانید وارد تنظیمات مرورگر خود شوید و در قسمت proxy اطلاعات اتصال را وارد کنید.

دقت کنید که تنها حالت HTTP را فعال کنید زیرا ما در این پروژه به دلیل سادگی کار تنها توانایی پاسخ به درخواست های رمز نشده HTTP را داریم.

بعد از پیاده سازی اولیه برنامه خود، مرورگر باید بتواند از proxy شما استفاده کند و ترافیک از برنامه شما عبور داده شود و کاربر از مرورگر استفاده کند، بعد از پیاده سازی کامل این قسمت ویژگی های دیگر برنامه را با توجه به راهنمایی آن قسمت به برنامه خود اضافه کنید.

## logging

شما در زمان اجرا باید یک فایل log در کنار خود ایجاد کند و تمامی اطلاعات را با ساعت وقوع در آن ذخیره کنید. اطلاعاتی مانند درخواست گرفته شده از مرورگر و جواب آن، درخواست فرستاده شده برای web server و جواب آن، و یا hit یا miss شدن cache باید در آن مشخص شود تا از درستی اجرای برنامه شما اطمینان حاصل شود و همچنین این کار در زمان debug کردن برنامه شما نیز می تواند کمکتان کند.

ابتدا از فایل config وضعیت این سرویس را بررسی کنید و در صورتی که سرویس فعال باشد باید در فایلی که مشخص شده است لاغ های خود را بنویسید. (اگه فایل وجود ندارد آن را بسازید و در صورتی که از قبل این فایل وجود داشته است در ادامه می آن شروع به نوشتن کنید)

نمونه فایل لاغ ضمیمه شده است و باید ساختاری شبیه به آن داشته باشد.

## Caching

باید در نظر داشته باشیم که فضای اختصاص داده شده برای Cache کردن پاسخ‌ها یک فضای محدود است. سیاست جایگزینی LRU می‌باشد. چون ذخیره کردن در فایل‌های سیستم مربوط به این درس نمی‌باشد نیازی به ذخیره اطلاعات پس از خارج شدن از پروکسی نمی‌باشد (نیازی به استفاده از دیتابیس نیست). باید توجه داشت که فضای محدود در اینجا به معنای حجم محدود نیست و محدودیت بر روی تعداد بسته‌های Cache شده می‌باشد. بدین معنا که مثلاً حداقل 100 پاسخ میتواند Cache شود. (این مقدار را باید از فایل config.json بخوانید)

### نکات مهم برای پیاده‌سازی بخش caching:

برای پیاده‌سازی این بخش دقت کنید که تمام پاسخ‌ها نباید cache شوند، بلکه باید با توجه به مقدار تنظیم شده در پرچم Pragma تصمیم‌گیری درستی در این مورد گرفته شود.

شما در این پرژوهه باید مقدارهای زیر را برای cache پشتیبانی کنید.

باید دقت داشت که در صورتی که پرچم proxy تنظیم نشده باشد میتواند این بسته را در Cache ذخیره کند.

## No-cache

این مقدار تنها روی بسته‌های پاسخ درج می‌شود و مقدار پرچم pragma می‌باشد. به این معناست که proxy نباید این بسته را در Cache ذخیره کند.

## :Expire

بسته تا این تاریخ اعتبار دارد و نیازی نیست برای پاسخ هر درخواست یک بسته به سرور استفاده کنیم و از همان مقدار موجود در Cache برای آن استفاده میکنیم.

## :If-Modified-Since

اگر شما بسته‌ای را cache کرده باشید و می‌خواهید اعتبار بسته را بررسی کنید میتوانید از این پرچم استفاده کنید، این پرچم در دو حالت در درخواست‌ها فرستاده می‌شود:

اگر در سرآیند، پرچم Expire تنظیم نشده باشد، در این صورت برای پاسخ هر درخواست که قبلاً در Cache ذخیره شده است باید یک بسته شامل این پرچم تنظیم شده باشد و به سرور فرستاده شود. در صورتی که جواب 304 از سمت سرور بازگردد، یعنی بسته تغییر نکرده است و همان مقدار موجود در Cache را به کلاینت ارسال میکنیم و اگر پاسخ 200 بازگردد باید مقدار موجود در Cache آپدیت شود و هم بسته جدید به کلاینت ارسال شود.

اگر بسته منقضی شده باشد (اگر از تاریخ Expire گذشته باشد) در صورت عدم استفاده از این پرچم دوباره درخواست گرفتن بسته به سرور فرستاده می‌شود و سرور دوباره بسته را ارسال می‌کرد.

در حالی که این پرچم این مشکل را حل کرده است. در این حالت یک بسته شامل این پرچم به سرور ارسال میکنیم و در صورتی که پاسخ 304 برگردانده شد برچسب زمان برای آن پاسخ بروزرسانی می‌شود. در صورتی که پاسخ 200 فرستاده شد (این پاسخ همراه با فایل فرستاده می‌شود) مقدار موجود در Cache را دوباره بروزرسانی میکنیم.

## Privacy

در زمان گرفتن درخواست از مرورگر، سیستم عامل و مرورگر در header با عنوان user-agent مشخص می‌شود. server می‌تواند این header را تغییر دهد و user-agent خودش را که در فایل config مشخص شده است بجای آن قرار دهد تا در زمان وبگردی حریم‌شخصی افراد رعایت شود.

این قسمت بسیار ساده است و تنها کافیست در صورتی که این سرویس فعال است، سرآیند user-agent در زمان ارسال را به مقدار داده شده در فایل config تغییر دهید و درخواست را برای وب‌서ور ارسال کنید.

## Notify the administrator by e-mail

یکی دیگر از مزیت های استفاده از پروکسی (برای مدیر سازمان یا صاحب پروکسی) کنترل و نظارت بر ترافیک عبوری از پروکسی می باشد. در صورتی که این سرویس فعال باشد (سروری Alert در فایل config.js) در صورتی که سرآمد host با یکی از ویسایت های هدف برابر بود باید بسته دریافت شده توسط پروکسی را با استفاده از Telnet و پروتکل SMTP برای مدیر سازمان ارسال کنید. بعد از ارسال بسته به ایمیل مدیر، پروکسی شما باید درخواست آن ها را به صورت کامل Drop کند و دسترسی کاربر بسته شود. (اتصال socket با مرورگر به صورت کامل بسته شود یا یک پیام اخطار نمایش داده شود)

### نحوه ارسال ایمیل:

برای ارسال ایمیل می توانید از سرور mail دانشگاه استفاده کنید. قبل از شروع می توانید با استفاده از telnet اقدام به ارسال ایمیل کنید، بعد از مطمئن شدن از درستی کار می توانید با استفاده از socket در زبان برنامه نویسی خود اقدام به ارسال ایمیل کنید.

برای پیدا کردن mail server میتوانید دستورات زیر را در ترمینال وارد کنید:

```
$ nslookup  
> set type=mx  
> <domain(e.g. ut.ac.ir)>
```

برای متصل شدن به سرور mail می توانید با دستور telnet mail.ut.ac.ir 25 در ترمینال اقدام به شروع کردن مکالمه با سرور کنید. برای معرفی خود از دستور زیر استفاده کنید:

HELO <name>

برای وارد کردن آدرس فرستنده

MAIL FROM: <"Sender e-mail address">

در این قسمت لازم است برای احراز هویت نام کاربری و رمز خود را وارد کنید:  
با دستور

AUTH LOGIN

برای احراز هویت اقدام کنید.

باید دقت داشته باشید که نام کاربری باید به فرمت Base64 باشد.

باید توجه داشته باشید که نام کاربری همراه با کاراکتر (\n) را به Base64 تبدیل کنید. مثلا:

admin\n : YWRtaW4K

برای تبدیل رمز به فرمت Base64 تنها خود رمز را به Base64 تبدیل کنید. مثلا:

admin: QWRtaW4=

در این مرحله باید آدرس گیرنده را وارد کنید:

RCPT TO: <Recipient e-mail address>

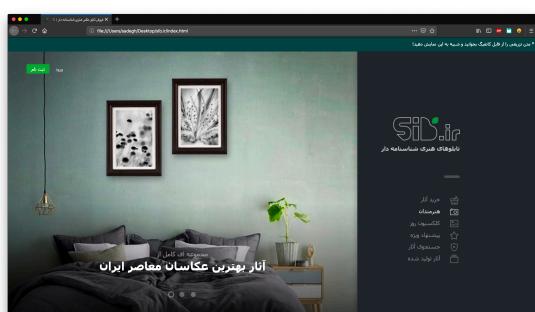
اگر با پیام تایید موافق شدید می توانید داده با زدن دستور زیر متن خود را برای سرور ایمیل کنید:

DATA

اگر با مشکلی روپروردید با جستجویی ساده در اینترنت می توانید آموزش های کامل تری بباید.

در این بخش حق استفاده از کتابخونه های آماده مانند SmtpLib و ... را ندارید و باید به صورت خام ایمیل را ارسال کنید.

## Http response injection



امروزه بسیاری از وب سایت ها از ارتباط این TLS استفاده می کنند که امکان شنود و تغییر در محتویات درخواست و جواب ها را از افرادی که در مسیر درخواست ها هستند می گیرد اما هنوز هستند وب سایت هایی که از پروتکل HTTP استفاده می کنند. در این قسمت از پروژه می خواهیم امکان تغییر جواب web server را به proxy خود اضافه کنیم.

برای این منظور هدف خود را تمام وب سایت هایی که از HTTPS استفاده نمی کنند قرار میدهیم، می خواهیم یک متن به Navbar اضافه کنیم،

برای این منظور اگر کاربر صفحه index یک سایت فاقد ارتباط امن را درخواست داده بود درخواست را برای web server سایت

هدف ارسال می‌کنیم ولی زمانی که جواب را دریافت کردیم با تغییر DOM باید یک Navbar به آن اضافه کنید و برای مرورگر ارسال کنید!

## Accounting

یکی دیگر از محدودیت‌هایی که می‌تواند بر روی proxy‌ها اعمال شود دادن حجم مشخص به هر کاربر می‌باشد. پروکسی شما باید تنها به ip‌هایی که در فایل config و در قسمت users آمده‌اند اجازه اتصال دهد. در ابتدای اجرای پروکسی، هر کاربر حجم ترافیک مشخصی دارد که پروکسی شما باید تنها به همان اندازه اجازه استفاده از ترافیک را به کاربر بدهد. مقدار شارژ هر فرد را در اجرای برنامه از فایل کانفیگ بخوانید، برای سادگی کار نیازی به persistant بودن شارژ کاربر نیست و نیاز به درگیر شدن با دیتابیس نمی‌باشد، می‌توانید این مقدار را در طول اجرای برنامه در ساختمان داده‌ای نگه‌داری کنید.

احراز هویت کاربران توسط ip آن‌ها انجام می‌شود؛ با دریافت هر بسته درخواست توسط پروکسی، در صورتی که ip درخواست دهنده در لیست کاربران پروکسی بود و حجم ترافیک او باقی ماند بود، اجازه استفاده از پروکسی را داراست و در غیر این صورت ارتباط بسته می‌شود یا می‌توانید پیامی مناسب برای کاربر نمایش دهید.

### منابع:

برای امتحان کردن حالت‌های مختلف Cache می‌توانید از این سایت‌ها کمک بگیرید:

<http://mydiba.xyz>  
<http://www.irna.ir>  
<http://sib.ir>  
<http://bdood.ir>  
<http://www.resaa.net>

برای مشاهده بسته‌های ارسال شده و دریافت شده می‌توانید از لینک زیر کمک بگیرید:

<https://websniffer.cc>

قبل از شروع بخش caching پروژه حتماً لینک‌های زیر را مطالعه کنید:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching>  
<https://www.keycdn.com/blog/http-cache-headers>

برای تبدیل نام کاربری به فرمت Base64 می‌توانید از سایت زیر کمک بگیرید.

<https://www.base64encode.org/enc/word>

### توضیحات تکمیلی (با دقت مطالعه کنید!)

- پروژه در گروه‌های دو نفره انجام می‌شود.
- تمیزی کد مهم است و بخشی از نمره را به خود تخصیص می‌دهد.
- برنامه شما باید درخواست‌ها را به صورت هم‌زمان پاسخ دهد. (می‌توانید از thread استفاده کنید)
- زمانی که درخواست را برای وب‌서ور ارسال می‌کنید باید شماره نسخه درخواست را به 1.0/HTTP/1.0 تغییر دهید.
- از هر زبانی می‌توانید استفاده کنید ولی کدهای شما باید سطح پایین و با استفاده از socket نوشته شوند.
- برای parse کردن درخواست‌های http استفاده از ماژول آماده مجاز نیست و باید به صورت دستی انجام شود. (برای json و html بلامانع است)
- استفاده از زبان‌هایی مانند C++ تنها به پیچیدگی کدهای شما می‌افزاید که از هدف پروژه به دور است، اکیداً پیشنهاد می‌شود از زبان python استفاده کنید.
- در صورت IO Async بودن پروکسی شما نمره امتیازی به شما تعلق می‌گیرد (برای مثال python asyncio را جستجو کنید)
- دقت کنید که تمام جزئیات مربوط به عمل کرد پروکسی شما باید در فایل log ثبت شوند.