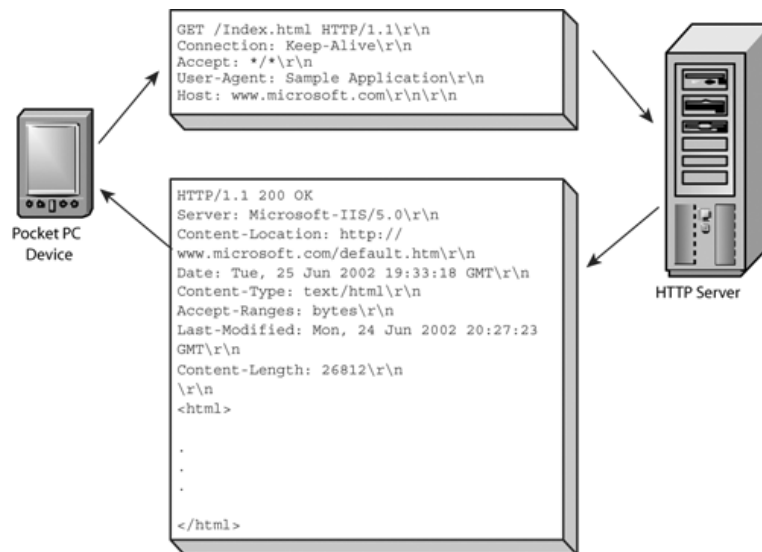


ساختار بسته های HTTP

پروتکل HTTP از یک ساختار متنی و نوشتاری که شامل کاراکترهای حروفی است، برای ارسال و دریافت بسته ها استفاده می کند. هر بسته ی HTTP، دارای یک سرآیند (Header) است و پس از آن، محتوای اصلی بسته درج می شوند. نمونه ای از یک پرسش و پاسخ تحت پروتکل HTTP در زیر آمده است:



می توان در یک Session ارتباط TCP، تعدادی زیادی پرسش و پاسخ رد و بدل شود، اما برای سادگی در مدیریت درخواست ها، در هر Session، یک درخواست ارسال می شود و به ازای آن نیز یک پاسخ دریافت می شود (Non-persistent HTTP) و سپس ارتباط قطع شده و برای درخواست بعدی، یک Session جدید باید ایجاد گردد.

در پاسخ یک درخواست از طرف سرور، به توجه به اطلاعات سرآیند آن می توان به اطلاعات مورد نیاز پروژه دست یافت. در اولین خط از پاسخ درخواست، پس از ذکر نام پروتکل HTTP و نسخه آن (معمولاً ۱،۱)، کد مربوط به وضعیت و توضیح وضعیت درج می گردد. در جدول زیر فهرست بعضی از کدهای تعریف شده آورده شده است:

HTTP Status Codes

Code	Description	Code	Description
200	OK	400	Bad Request
201	Created	401	Unauthorized
202	Accepted	403	Forbidden
301	Moved Permanently	404	Not Found
303	See Other	410	Gone
304	Not Modified	500	Internal Server Error
307	Temporary Redirect	503	Service Unavailable

لذا در صورت دریافت کد 200 از طرف سرور، به معنای موفقیت آمیز بودن درخواست و پاسخ آن است. اما اگر کد های 301 یا 302 از طرف سرور دریافت شده، به معنای جابجایی محتوای مورد نظر به آدرس دیگری است. کد 301 به معنای جابجایی دائمی و کد 302 به معنای جابجایی موقتی است که تفاوت این دو برای مرورگر فرقی ندارد و تنها برای موتورهای جستجو کاربردی است. لذا در صورت دریافت این کد ها، در Header یک فیلد به نام Location، محل جدید فایل مورد نظر را معرفی می کند تا مرورگر بتواند به آن دسترسی پیدا کند. همچنین در صورت دریافت کد 404، به معنای عدم موجود بودن آن محتوا در آدرس مورد نظر است. در صورتی که کد 200 از طرف سرور دریافت شود، محتوای اصلی صفحه در پایین Header درج می گردد. به طول کلی دو نوع مختلف برای درج اطلاعات در قسمت بدنه ی پروتکل HTTP وجود دارد. یکی از این روش ها طول ثابت و استفاده از فیلد Content-Length است و نوع دیگر استفاده از Transfer-Encoding از نوع Chunked است. در روش اول، طول کل محتوا مورد نظر، در Header درج می شود.

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

اما در روش دوم، در قسمت بدنه اصلی، محتوا به بخش های کوچک (Chunk) شکسته می شود و قبل از شروع هر بخش، طول آن بخش ذکر شده است. پس ابتدای بدنه ی اصلی با یک عدد آغاز می شود که بیانگر حجم اطلاعات موجود در قسمت اول است و پس از پایان قسمت اول، طول قسمت بعدی ذکر می شود. لذا به این ترتیب می توان به محتوای ارسال شده دسترسی پیدا کرد. همچنین در پایان بدنه ی پروتکل، با ذکر عدد صفر به معنای قسمت خالی، پایان محتوای دریافت شده اعلام می شود.

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Encoding: gzip
Transfer-Encoding: chunked

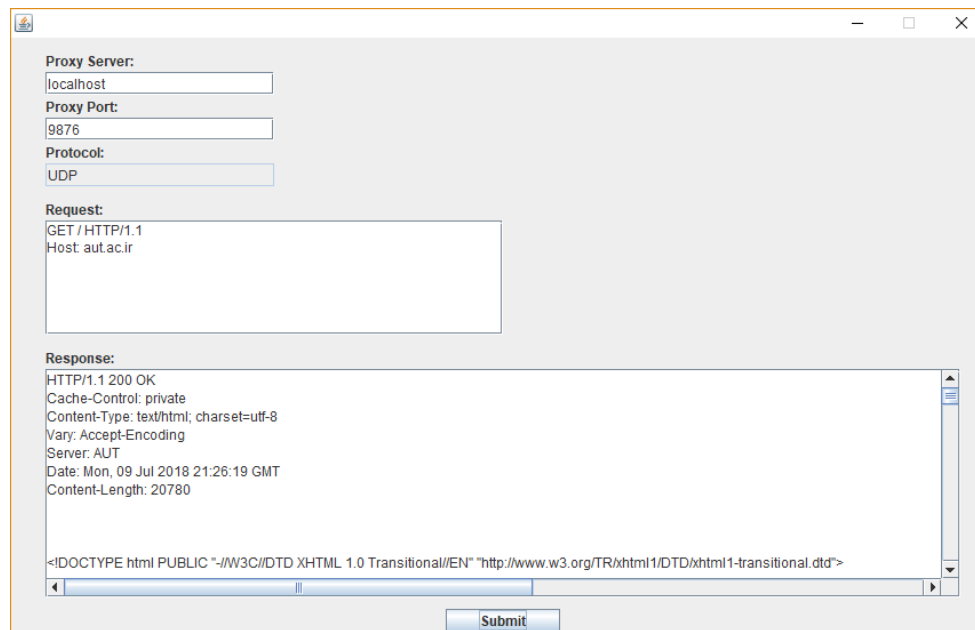
2C
šŃŹ<+h±¼ðf""Zpeé+Œ<DCmÃ°7h_fiuë+àì!Bõ×5"i£Ÿ
1E
«Ů«Ōă@J®³©v*í©ŮŸŸ^Œçxý-3&±,µž
0

```

CRLF
 response line
 headers
 compressed
 chunked
 start of body
 length
 chunk 1
 length
 chunk 2
 length
 end of body

نرم افزار WebBrowser

نرم افزار WebBrowser به صورت گرافیکی و با استفاده از کتابخانه Swing جاوا طراحی شده است، به گونه ای عمل می کند که با وارد کردن آدرس IP و Port پروکسی و همچنین متن درخواست HTTP و فشردن دکمه ی Submit، درخواست را به صورت UDP به پروکسی سرور ارسال می کند تا پروکسی جواب را به صورت TCP از مقصد نهایی گرفته و به صورت UDP به WebBrowser ارسال نماید. پاسخ از طرف سرور نیز به صورت کاملاً Raw و پردازش نشده در فیلد Response نمایش داده می شود. نمایی از برنامه ی ساخته شده به صورت زیر است:



درخواست ها به صورت ساده و همانطور که در صورت مسئله ی پروژه ذکر شده است ارسال می شوند. تنها عملیاتی که نرم افزار بر روی متن درخواست انجام می دهد، بررسی این موضوع است که آیا درخواست با کاراکتر های "\r\n\r\n" خاتمه می یابد یا خیر و در صورت وجود اشکال، آن را تصحیح می کند و سپس برای پروکسی ارسال می کند تا به ایرادهای بعدی برخورد نکند. اما برای پاسخ ها احتیاج به تحلیل سرآیند و محتوا وجود دارد. در صورت مواجه شدن با پیغام های 301 و 302، WebBrowser با استفاده از آدرس مشخص شده در فیلد Location، درخواست جدیدی را برای پروکسی سرور ارسال می کند. به عنوان مثال اگر فیلد Location به صورت زیر باشد:

Location: http://www.mywebsite.net/myfolder/mypage/

در خواستی به صورت زیر مجدداً برای پروکسی ارسال می گردد:

GET /myfolder/mypage/ HTTP/1.1

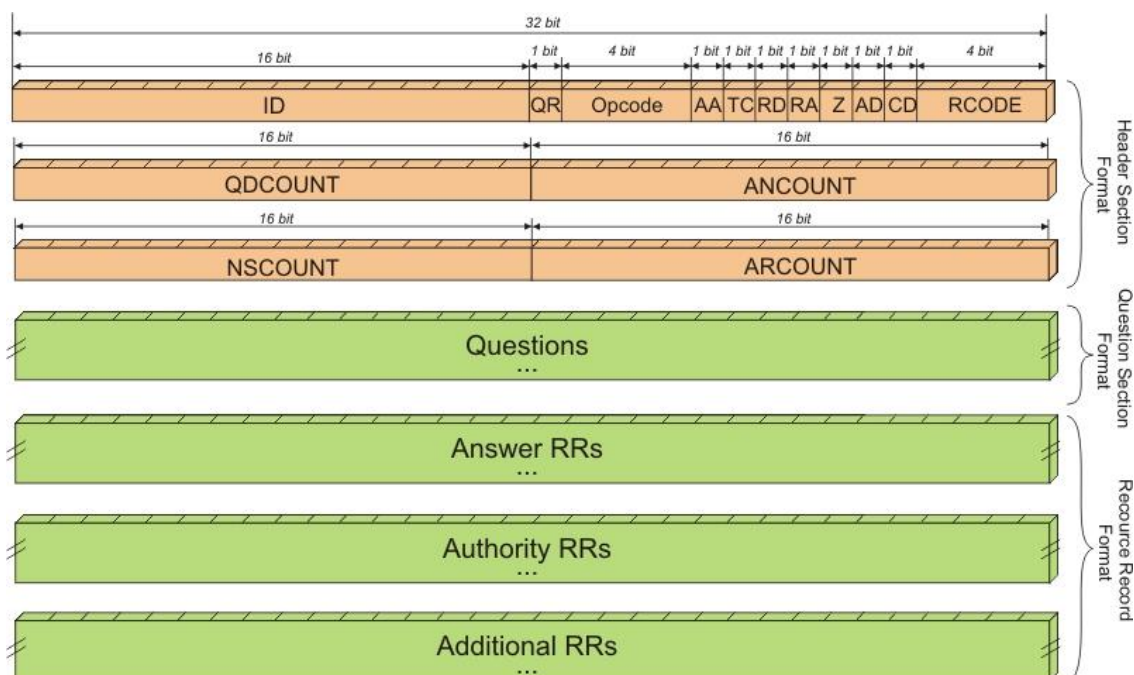
Host: www.mywebsite.net

تا جایی که WebBrowser به کد پاسخ 200 دسترسی پیدا کند، این عملیات تکرار خواهد شد. در صورت دریافت کد 200، عملیات جداسازی Header و آماده سازی داده با استفاده از روش های Content-Length یا Chunked آغاز می شود و فایل

نهایی تحت عنوان output.html ایجاد می‌گردد. همچنین کد 404 به معنای عدم یافتن محتوای مورد نظر است و بدون هیچ بررسی‌ای، بسته دور ریخته می‌شود. در صورت اینکه پیغام "Website not found!" از طرف پروکسی دریافت گردد، به این معنی است که Host وارد شده در درخواست وجود خارجی ندارد و در صورت دریافت کدهایی دیگری غیر از خطاهای ذکر شده در بالا، پیغام "Unkown Error!" ظاهر می‌گردد.

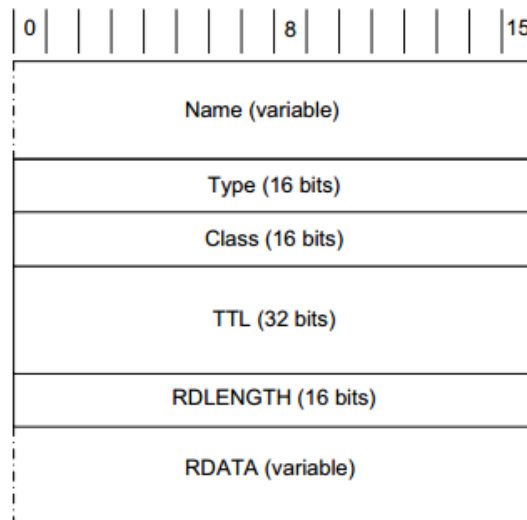
ساختار بسته های DNS

ساختار درخواست‌ها و پاسخ‌های DNS مشابه بوده و از یک فرم برای Query و Response استفاده می‌گردد. این ساختار دارای یک قسمت سرآیند است و در ادامه آن، داده‌ها قرار می‌گیرد که به فرم زیر است:

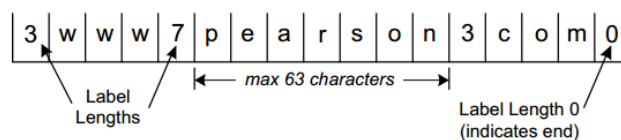


در ابتدا ۲ بایت ID قرار می‌گیرد که به عنوان شماره درخواست استفاده می‌گردد و یک عدد کاملاً تصادفی است. ۲ بایت بعدی مربوط به Flag ها می‌باشد که اطلاعاتی نظیر مشخص کردن اینکه این پیغام یک Query یا Response است، Authority بودن سرور مقصد برای دامنه مورد نظر، وضعیت پیغام از نظر اینکه درخواست موفقیت آمیز بوده است و یا دامنه مورد در لیست DNS سرور مقصد موجود نبوده است را در خود جای می‌دهد که در ادامه برای تحلیل این بسته‌ها به کار می‌رود.

در ۲ بایت بعدی تعداد رکورد های Query و سپس در ۲ بایت بعدی تعداد رکورد های Answer و در ۲ بایت بعدی تعداد رکورد های Authority و در ۲ بایت بعدی نیز رکورد های Additional قرار می‌گیرند که با توجه به تعداد و ترتیب آن‌ها، این داده‌ها در ادامه‌ی Header قرار می‌گیرند. هر کدام رکوردهای این ۴ بخش دارای ساختارها و فیلدهای مربوط به خود هستند اما دارای یک قسمت مشترک به صورت زیر هستند که در تمامی این ۴ بخش تقریباً یکسان است و در هر بخش فیلدهای دیگری علاوه بر این فیلدها به رکوردها اضافه می‌گردد:



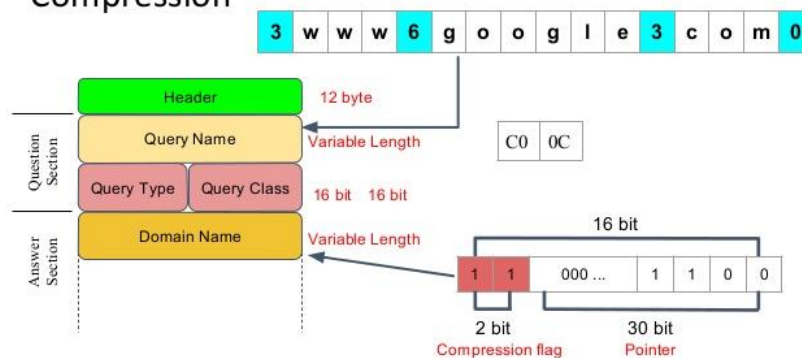
در ابتدا هر رکورد، نام رکورد که در واقع همان نام دامنه است ذکر می‌گردد. نام دامنه تنها فیلد دارای طول متغیر در بین رکوردها است و سایر فیلدها دارای تعداد بایت مشخص هستند. نام دامنه به دو صورت کامل و یا فشرده (با استفاده از اشاره‌گر) در اول هر رکورد درج می‌شود. به عنوان مثال برای دامنه www.pearson.com، شکل ذخیره سازی آن به صورت زیر است:



هر بخش از نام که با نقطه جدا می‌شود، حاوی تعدادی کاراکتر است. در حالت کامل، به جای ذخیره سازی نقطه‌ها در فیلد نام رکورد، در ابتدای هر بخش از نام، تعداد کاراکترهایی که در ادامه آن می‌آید ذکر می‌شود. مثلاً دارای ۳ کاراکتر است و **pearson** دارای ۷ کاراکتر و **com** دارای ۳ کاراکتر است و در صورت رسیدن به عدد صفر، به معنای اتمام فیلد نام است.

اما در حالت فشرده، در صورت اینکه ۲ بیت اول هر بخش، یک باشند، ۳۰ بیت بعدی به عنوان اشاره گر به کار می‌رود. به این صورت که اگر قبلاً این نام از دامنه در پیغام ذکر شده است، با ارجاع به آن که در چندمین بایت از پیغام قرار دارد، از ذکر کامل آن نام صرف‌نظر می‌کنیم:

Compression

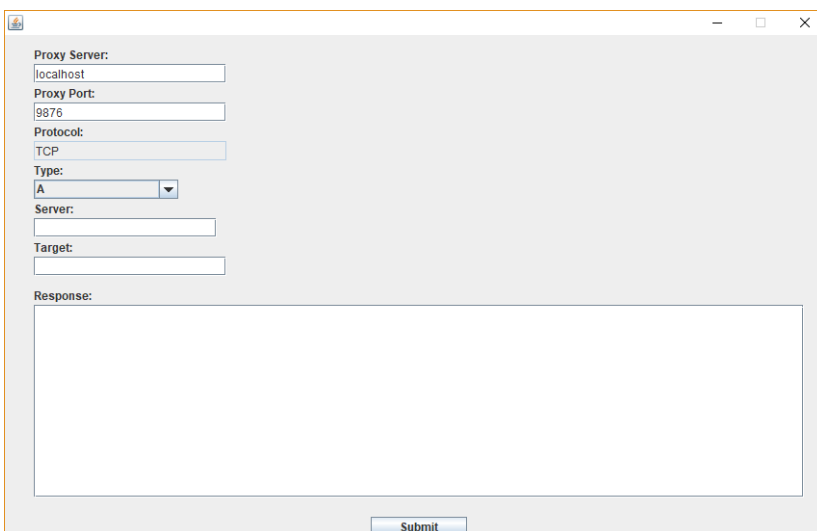


این دو روش به صورت ترکیبی نیز به کار می‌روند، به عنوان مثال اگر دامنه ی `shop.test.com` قبلاً در پیغام ذکر شده است و بخواهیم دامنه ی `new.shop.test.com` را در پیغام ذخیره کنیم، تنها بخش اول اسم را ذخیره کرده و بخش های بعدی را با استفاده از اشاره گر، ارجاع می‌دهیم.

پس از فیلد نام، ۲ بایت فیلد `Type` است که نوع آن رکورد (`A`، `CNAME` و ...) را مشخص می‌کند. همچنین ۲ بایت مربوط به فیلد `Class` نیز در ادامه می‌آید که معمولاً کلاس رکورد از نوع `IN` یا `Internet` است و به همین ترتیب سایر فیلدها شامل طول داده (که برای `IPv4` دارای مقدار ۴ است) و خود `IP` مورد نظر در هر رکورد ذکر می‌شوند.

نرم افزار NSlookup

این نرم افزار مشابه نرم افزار `WebBrowser`، در ابتدا مشخصات `IP` و `Port` پروکسی سرور را دریافت می‌کند. پروتکل پیشفرض برای ارتباط بین نرم‌افزار و پروکسی `TCP` است. سپس با دریافت اطلاعات مربوط به `Type`، `Server` و `Target` و فشردن دکمه ی `Submit`، یک ارتباط `TCP` بین نرم‌افزار و پروکسی برقرار می‌گردد.



The screenshot shows a window titled 'NSlookup' with a light gray background. It contains several input fields and a button. The fields are labeled: 'Proxy Server:' with 'localhost' entered, 'Proxy Port:' with '9876' entered, 'Protocol:' with 'TCP' entered, 'Type:' with a dropdown menu showing 'A', 'Server:', and 'Target:'. Below these is a large empty text area labeled 'Response:'. At the bottom right is a button labeled 'Submit'.

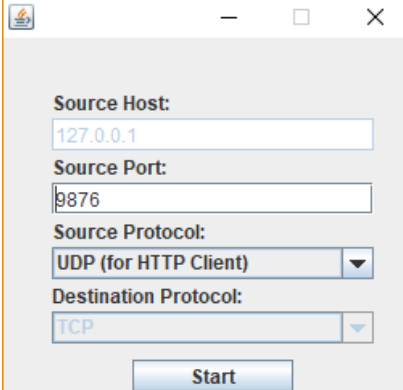
یک بسته به صورت زیر ایجاد شده و برای پروکسی ارسال می‌گردد:

`type=[TYPE] server=[SERVER_ADDRESS] target=[TARGET_NAME]`

و پس از آنکه پروکسی پاسخ را از `DNS` سرور مقصد دریافت کرد، سپس پاسخ را با همان فرمت استاندارد `DNS` که پیش تر تشریح شد، به صورت `TCP` برای نرم‌افزار ارسال می‌کند. نرم‌افزار با تحلیل بیت‌های `Flag` به `Authority` بودن سرور مقصد پی می‌برد. همچنین آدرس `IP` مربوط به دامنه را نیز از بسته استخراج می‌کند و برای پاسخ پرسش های نوع `CNAME` که شامل فیلد `primaryNameServer` است، اطلاعات مربوط به نام دامنه‌ی اصلی را استخراج کرده و مجموع این اطلاعات را در قسمت `Response` برای کاربر نمایش می‌دهد.

نرم افزار پروکسی ProxyProject

این نرم افزار دارای دو بخش کلی UDPToTCP و TCPToUDP است که یکی برای کار با نرم افزار NSLookup و دیگری برای WebBrowser ساخته شده است. در ابتدا از کاربر پرسیده می شود که پروکسی بر روی کدام پورت و با کدام پروتکل ایجاد گردد:



The screenshot shows a small application window titled 'ProxyProject'. It contains four input fields and two dropdown menus. The 'Source Host' field is set to '127.0.0.1'. The 'Source Port' field is set to '9876'. The 'Source Protocol' dropdown menu is set to 'UDP (for HTTP Client)'. The 'Destination Protocol' dropdown menu is set to 'TCP'. At the bottom of the window is a 'Start' button.

و پس از زدن دکمه Start بر روی پورت مورد نظر، listen می کند.

در حالت UDPToTCP، پس از دریافت یک درخواست HTTP به صورت UDP از کلاینت، با استخراج فیلد Host از درخواست، مقصد ارسال بسته را تعیین می کند و با ایجاد ارتباط TCP با آن سرور و فرستادن درخواست به همان فرم و جمع آوری پاسخ، آن را به صورت UDP برای کلاینت، ارسال می کند. بسته ها در اندازه های ۱۰۰۰ بایتی برای کلاینت ارسال می شود و در صورت پایان یافتن اطلاعات، با ارسال یه بسته UDP با طول صفر، پایان آن را اعلام می کند.

در حالت TCPToUDP، پس از دریافت یک پرسش DNS به صورت TCP از کلاینت، با استخراج فیلد server از درخواست، مقصد ارسال بسته را تعیین می کند و با ایجاد ارتباط UDP با آن سرور و فرستادن درخواست به فرم استاندارد DNS (مطابق با توضیحات بالا) و جمع آوری پاسخ، آن را به صورت TCP برای کلاینت، ارسال می کند و به ارتباط TCP خود با کلاینت پایان داده و دوباره بر روی پورت تنظیم شده listen می کند.

در طراحی این نرم افزار از مکانیزم های Cache نیز استفاده شده است. هر رکورد Cache که دارای کلاس CacheEntry است، شامل پاسخ و تاریخ انقضا می باشد که تاریخ انقضا با استفاده از کتابخانه های تاریخ و تقویم جاوا ذخیره می شود. همچنین یک ساختار Map که بدنه ی اصلی Cache را تشکیل می دهد ایجاد شده است. این ساختار هر درخواست را به یک CacheEntry متناظر می کند. پس از دریافت یک Query، ابتدا کل ساختار Cache بررسی می شود و تمام داده هایی که تاریخ انقضا آن ها فرا رسیده است، دور ریخته می شود. سپس درخواست دریافت شده بررسی می شود که در Cache موجود است یا خیر. در صورتی که موجود بود، با استفاده از تناظر یک به یک ساختار Map، پاسخ متناظر آن به دست آمده و برای کلاینت ارسال می گردد. اما در غیر این صورت، درخواست به مقصد ارسال می گردد و پاسخ آن علاوه بر اینکه برای کلاینت ارسال می گردد، در حافظه ی Cache نیز با تاریخ انقضا مشخص، ذخیره می گردد.