

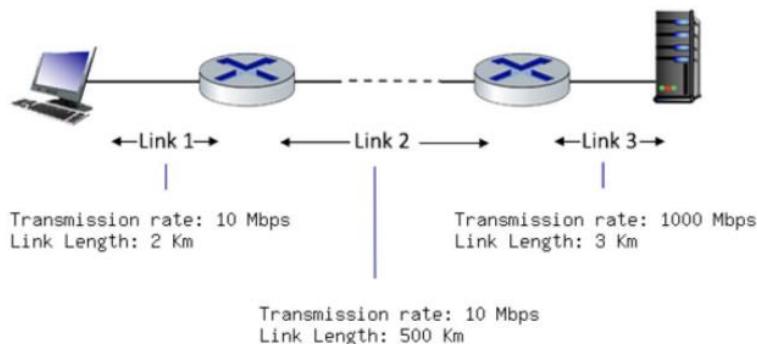
# Computer Network

**AmirReza Azari**  
**99101087**

## تمرین اول

### سوال اول

سؤال ۱ با توجه به شکل زیر به سوالات زیر پاسخ دهید. سرعت انتشار سیگنال روی لینک‌ها را  $2 \times 10^8 \text{ m/s}$  است.



- (آ) با فرض ناچیز بودن زمان پردازش و زمان معطلی در صفحه، یک بسته  $B = 1500B$  بعد از چند ثانیه از *server* به *client* می‌رسد؟
- (ب) اگر بخواهیم فایلی به طول  $15000B$  را به بسته‌های  $1500B$  تقسیم کرده و همه را پشت سر هم از *server* به *client* ارسال کنیم، با فرض ناچیز بودن زمان پردازش، کل فایل در چه مدت زمانی به سرور می‌رسد؟
- (ج) حداکثر گذردهی (*Throughput*) در حالت ب قدر است؟ چرا؟
- (د) اگر بخواهیم زمان رسیدن فایل در حالت ب را به زیر  $5ms$  برسانیم، چه راه حلی پیشنهاد می‌کنید؟

### پاسخ سوال اول:

(آ)

می‌دانیم:

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

$d_{\text{trans}}$ : transmission delay:

- $L$ : packet length (bits)
- $R$ : link transmission rate (bps)
- $d_{\text{trans}} = L/R$

$d_{\text{prop}}$ : propagation delay:

- $d$ : length of physical link
- $s$ : propagation speed ( $\sim 2 \times 10^8 \text{ m/s}$ )
- $d_{\text{prop}} = d/s$

$d_{\text{trans}}$  and  $d_{\text{prop}}$   
very different

طبق گفته صورت سوال،  $d_{proc} \approx d_{queue} \approx 0$  تا خیر دیگر را مطابق فرمول بالا محاسبه نماییم.

$$D_{transe} = \frac{L}{R1} + \frac{L}{R2} + \frac{L}{R3}$$

$$D_{prop} = \frac{d1}{s} + \frac{d2}{s} + \frac{d3}{s}$$

$$L = 1500 * 8 = 12000 \text{ bit}$$

حال محاسبات را انجام می‌دهیم:

$$D_{transe} = \frac{12000}{10000000} + \frac{12000}{10000000} + \frac{12000}{1000000000} = 0.002412 s$$

$$D_{prop} = \frac{2000}{2 \times 10^8} + \frac{500000}{2 \times 10^8} + \frac{3000}{2 \times 10^8} = 0.002525 s$$

$$D_{total} = D_{transe} + D_{prop} = 0.002412 + 0.002525 = 0.004937 s = 4.937 ms$$

بنابراین جواب این بخش برابر با 4.937 میلی ثانیه می‌باشد.

(ب)

10 بسته داریم که پشت سر هم ارسال خواهند شد. بسته اول بعد از 4.937 میلی ثانیه خواهند رسید و هر بسته بعد از آن، پس از  $\frac{L}{R_{min}}$  بعد از بسته پیش از خود خواهد رسید. بنابراین داریم:

$$\frac{L}{R_{min}} = \frac{1500 \times 8}{10 \times 10^6} = 0.0012 s = 1.2 ms$$

$$Total delay = 4.937 + 9 \times 1.2 = 15.737 ms$$

بنابراین جواب این بخش برابر با 15.737 میلی ثانیه می‌باشد.

(ج)

برابر minimum مقدار بین تمامی transmission rate ها است. بنابراین 10Mbps، به دلیل گلوگاه بودن لینک چپ و وسط

(د)

10 برابر کردن سرعت دو لینک کندتر، زیرا:

$$\begin{aligned} \text{delay of first packet} &= d_{trans} + d_{prop} \\ &= \left( 1500 \times 8 \times \left( \frac{1}{100 \times 10^6} + \frac{1}{100 \times 10^6} + \frac{1}{1000 \times 10^6} \right) \right. \\ &\quad \left. + \frac{1}{2 \times 10^8} \times (2 + 500 + 3) \times 10^3 \right) = 0.000252 + 0.002525 \\ &= 0.002777 \text{ s} = 2.777 \text{ ms} \end{aligned}$$

و همچنین مقدار تاخیر هر بسته، بعد از بسته پیشین خود برابر است با:

$$\frac{L}{R_{\min}} = \frac{1500 \times 8}{100 \times 10^6} = 0.00012 \text{ s} = 0.12 \text{ ms}$$

و در نهایت جواب تغییر یافته بخش ب برای این قسمت برابر می‌شود با:

$$2.777 + 9 \times 0.12 = 3.857 \text{ ms}$$

که به خواسته سوال یعنی زیر 5 ms بودن رسیدیم.

## سوال دوم

سؤال ۲ HTTP یک پروتکل بدون حالت (state-less) است. با این حال در طول روز با سایت‌هایی رویه‌رو می‌شویم که حالت هر client را به گونه‌ای حفظ می‌کنند؛ به طور مثال با یک بار login دیگر نیاز به ورود دوباره نیست، یا سایت‌های فروشگاهی‌ای که بدون login کردن هم سبد خرید شما را حتی بعد از refresh نگه می‌دارند. توضیح دهید این سایت‌ها از چه مکانیزمی برای state-full کردن این پروتکل استفاده می‌کنند.

## پاسخ سوال دوم:

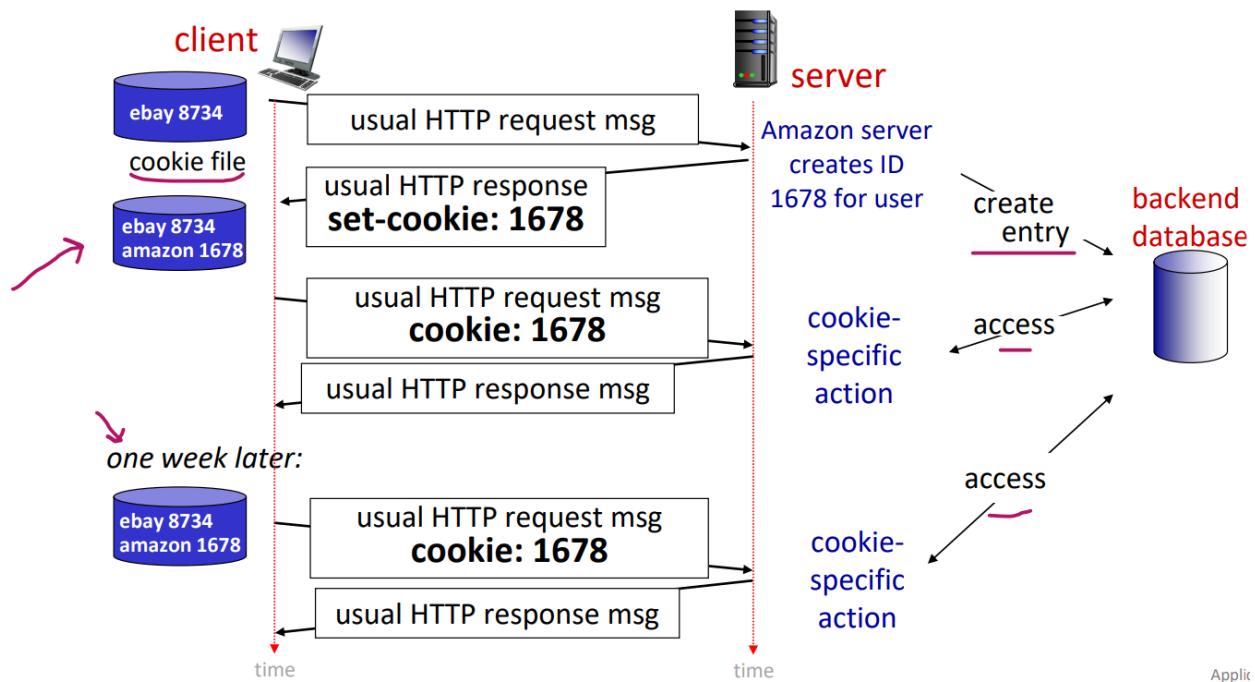
این پروتکل از cookies و sessions برای state-full کردن بهره می‌بریم. این دو روشی برای ذخیره و انتقال داده بین سرور و کلاینت می‌باشد و در ادامه هر کدام را توضیح خواهیم داد.

ابتدا cookies را شرح می‌دهیم:

کوکی‌ها قطعات کوچکی از داده‌ها هستند که توسط سرور بر روی دستگاه کلاینت ذخیره می‌شوند و با هر درخواست به سرور ارسال می‌شوند. کوکی‌ها می‌توانند شامل اطلاعات مختلفی درباره هویت کاربر، ترجیحات، تاریخچه و غیره باشند. کوکی‌ها می‌توانند با امكان تشخیص و به خاطر سپردن کاربر در طول چندین درخواست

و پاسخ، وضعیت HTTP را بازتاب دهند. به طور دیگر می‌توان اینگونه آنها را توضیح داد. کوکی‌ها فایل‌های متنی با قطعات کوچکی از داده‌ها مانند نام کاربری و رمز عبور هستند که برای شناسایی کامپیوتر در حال استفاده از یک شبکه استفاده می‌شوند. کوکی‌های خاص برای شناسایی کاربران خاص و بهبود تجربه مرور وب آنها استفاده می‌شوند. داده‌های ذخیره شده در یک کوکی توسط سرور در هنگام اتصال شما ایجاد می‌شود. این داده با یک شناسه منحصر به فرد برای شما و کامپیوتر شما برچسب‌گذاری می‌شود. زمانی که کوکی بین کامپیوتر شما و سرور شبکه تبادل می‌شود، سرور شناسه را می‌خواند و می‌داند کدام اطلاعات را به طور خاص برای شما ارائه دهد. در اولین پیام رد و بدل شده بین ما و سایت در response یک پیام HTTP کوکی به ما داده شود و آن را ست می‌نماید. شکل زیر از اسلایدها توضیح مختصری از نحوه عملکرد آنها می‌دهد.

## Maintaining user/server state: cookies



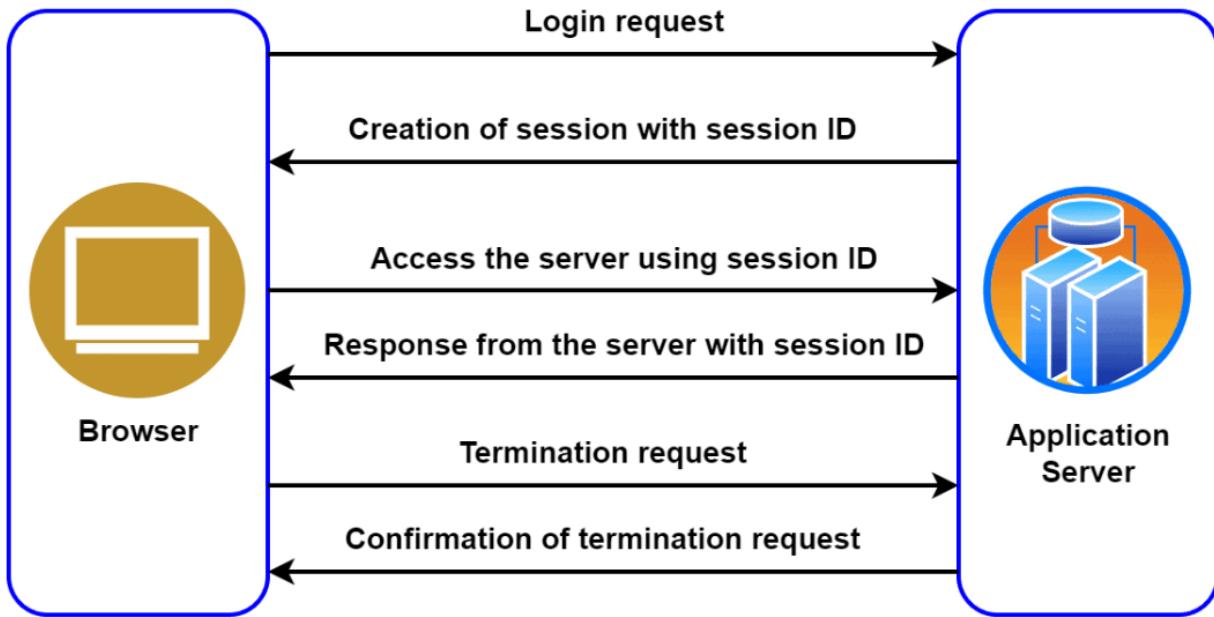
همچنین تصویر زیر ارسال و header کوکی را نشان می‌دهد.

[https://www.google.com/url?sa=i&url=https%3A%2F%2Fweblog.west-wind.com%2Fposts%2F2012%2FNov%2F29%2FSetCookie-Headers-getting-stripped-in-ASPNET-HttpHandlers&psig=AOvVawOQFNj2LxvjwDY1uCUOJyyO&ust=171268528345000&source=images&cd=vfe&opi=89978449&ved=0CBQQjhxqFwoTCNDX\\_ICYs4UDFQAAAAAdAAAABAE](https://www.google.com/url?sa=i&url=https%3A%2F%2Fweblog.west-wind.com%2Fposts%2F2012%2FNov%2F29%2FSetCookie-Headers-getting-stripped-in-ASPNET-HttpHandlers&psig=AOvVawOQFNj2LxvjwDY1uCUOJyyO&ust=171268528345000&source=images&cd=vfe&opi=89978449&ved=0CBQQjhxqFwoTCNDX_ICYs4UDFQAAAAAdAAAABAE)

The screenshot shows the Network tab of the Chrome Developer Tools. The Headers section is selected. A red oval highlights the 'Set-Cookie' header, which has the value 'WWTHREADSID=ThisIsTheValue; path=/'. Other visible headers include Host, Proxy-Connection, Referer, User-Agent, Cache-Control, Content-Length, Content-Type, Date, RequestId, and X-AspNet-Version.

و اما session نشستها (Sessions) فضاهای ذخیره‌سازی موقت هستند که برای هر کاربری که به یک برنامه وب دسترسی دارد، بر روی سرور ایجاد می‌شوند. نشستها می‌توانند شامل اطلاعات مختلفی درباره فعالیت، داده‌ها، وضعیت و غیره کاربر باشند. نشستها می‌توانند باعث ایجاد حالت متغیر در HTTP شوند، اجازه می‌دهند تا سرور تعامل کاربر با یک برنامه وب را در طول چندین درخواست و پاسخ‌شناسایی و مدیریت کند.

به طور کلی، کوکی‌ها برای ذخیره اطلاعات در دستگاه مشتری و نشستها برای ذخیره اطلاعات در سرور استفاده می‌شوند.



### سوال سوم

سؤال ۳ یک کاربر میخواهد از یک صفحه وب که شامل یک فایل *HTML* و 10 آجکت است بازدید کند. فایل *HTML* 5000 byte بوده و به 10 آجکت اشاره می‌کند، که دو تای آنها (01, 02) روی وبسرور 1 هستند که فایل *HTML* نیز روی آن قرار دارد، و 8 تای دیگر (03-010) روی وبسرور 2 هستند.

حجم فایل‌ها به این صورت است: 01=1000 byte, 02=3000 byte, 03-08 = 5000 byte, 09=010=2000 byte متوسط گذردهی لینک بین کامپیوتر و وبسرور 1 100000 bits/sec، و این مقدار برای لینک بین کامپیوتر و وبسرور 2 400000 bits/sec می‌باشد. برای سادگی، فرض کنید این مقدار مناسب با سایز فایل ارسالی بین ارتباطات موازی تقسیم می‌شود. همچنین برای وبسرور 1 داریم  $RTT_1 = 0.01s$  و برای وبسرور 2،  $RTT_2 = 0.02s$  و برای DNS داریم  $RTT_{DNS} = 0.005s$ . با توجه به اینکه کاربر در ابتدا آدرس هیچکدام از وبسرورها را ندارد، توضیح دهید چند میلی ثانیه طول می‌کشد تا کاربر با استفاده از مرورگری که با هر وبسرور، حداقل 5 ارتباط موازی *HTTP* از نوع persistent برقرار می‌کند، این صفحه وب را به صورت کامل دریافت کند.

### پاسخ سوال سوم:

برای حل این سوال بخش‌های زیر را طی می‌نماییم.

ابتدا باید ip address وب سرور اول را در 0.005 ثانیه بدست بیاوریم. سپس TCP connection که برابر با  $RTT_1 = 0.01$  ثانیه می‌باشد، داریم. حال باید خود فایل *html* را دریافت نماییم. برای این کار همان مقدار  $RTT_1 = 0.01$  ثانیه به علاوه میزان زمان ارسال فایل 5000 بایتی خواهیم داشت. بنابراین از ابتدا تا زمان دریافت فایل *html* داریم:

$$RTT_{DNS} + RTT_1 \text{ TCP connection} + RTT_1 + \text{time to transmit file}$$

$$= 0.005 + 0.01 + 0.01 + \frac{5000 \times 8}{100000} = 0.425 s$$

حال به سراغ دریافت آبجکت‌ها می‌رویم. طبق نکات ذکر شده در کوئرای درس (گفته شده «با هر وب سرور، حداقل ۵ ارتباط موازی» و منظور این هست که **همزمان** ۵ تا با سرور ۱ و ۵ تا با سرور ۲)، در این بخش ۲ کار را موازی انجام می‌دهیم. یکی دریافت دو آبجکت از وب سرور ۱ و دیگری دریافت ip address آدرس وب سرور دوم و دریافت ۸ آبجکت آن که ۵ تای اول آن با هم، و ۳ تای آخر نیز با هم انجام خواهند شد.

برای بخش اول داریم:

$$RTT_1 + \text{time to transmit files} = 0.01 + \left( \frac{1000 \times 8}{100000} + \frac{3000 \times 8}{100000} \right) = 0.33 s$$

در این بخش می‌توانیم فرض کنیم یک کانکشن دیگر باز کرده و ۰.۰۱ دیگر نیز هزینه می‌دهیم. از طرفی طبق نکات مطرح شده در کوئرای این نکته: "زودترین موقعی که می‌شه درخواست DNS برای وب سرور ۲ رو بزنیم می‌شه بعد از دریافت کامل فایل html و همزمان با دانلود فایل‌هایی که روی وب‌سرور ۱ هستن." می‌توانیم همین حالت را هم در نظر بگیریم که خواهیم دید در جواب نهایی بی‌تأثیر خواهد بود و همچنین با توجه به فرض persistent بودن، همین حالت به نظر منطقی می‌آید.

حال در بخش موازی دیگر داریم:

$$RTT_{DNS} + RTT_2 \text{ for TCP connection} + \text{one } RTT_2 \text{ for objects [03 – 07]}$$

$$+ \text{time to transmit files 03 – 07}$$

$$+ \text{one } RTT_2 \text{ for objects [08 – 10]} + \text{time to transmit files 08 – 10}$$

$$= 0.005 + 0.02 + 0.02 + \left( \frac{5 \times 5000 \times 8}{400000} \right) + 0.02$$

$$+ \left( \left( \frac{5000 \times 8}{400000} \right) + \frac{2 \times 2000 \times 8}{400000} \right)$$

$$= 0.005 + 0.02 + 0.02 + 0.5 + 0.02 + 0.1 + 0.08 = 0.745 s$$

حال می‌دانیم دو عملیات بالا موازی انجام شده‌اند. بنابراین ماکس زمان آن‌ها اثر گذار است. در نهایت داریم:

$$0.425 + \max(0.33, 0.745) = 1.17 s = 1170 ms$$

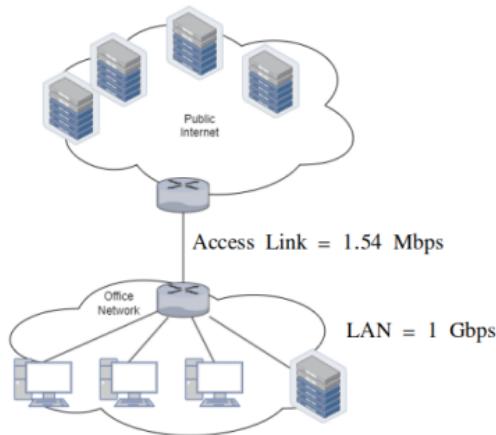
## سوال چهارم

سوال ۴ شبکه اینترنت یک شرکت به شکل زیر را در نظر بگیرید. کاربران شبکه از سرعت پایین اینترنت شاکی هستند. درباره درخواست‌های آنان، اطلاعات زیر در دسترس است:

- average request size = 2 Mb
- average request rate = 5 per sec
- Institutional router to origin server RTT = 2 sec

(آ) با فرض اینکه بتوان لینک متصل‌کننده به اینترنت عمومی را تعویض کرد، حساب کنید با 10 برابر شدن پنهای باند این لینک، Access Link Utilization چطور می‌کند؟

(ب) یک متخصص شبکه، پیشنهاد داده است به جای تغییر لینک، از یک web cache در شبکه استفاده شود. با توجه به اینکه hit-rate برای این cache حدوداً 40% خواهد بود، حساب کنید این روش قبل یا خیر؟ اگر بهتر نیست، به ازای چه hit-rate‌ای این روش بهتر عمل خواهد کرد و اگر بهتر است، به ازای چه hit-rate‌ای روش قبل بهتر عمل خواهد کرد؟ (برای مقایسه عملکرد در این حالت، Access Link Utilization را مقایسه کنید).



(ج) برای حالت اولیه و هر یک از حالات بالا، مدت تاخیر در پاسخ را حساب کنید. چه نتیجه‌ای می‌توان گرفت؟

### پاسخ سوال چهارم:

(الف)

قبل از 10 برابر کردن:

$$\text{avg data rate to browsers} = \text{average request size} \times \text{average request rate} = 2 \times 5 = 10 \text{ Mbps}$$

$$\text{Access Link Utilization} = \min\left(1, \frac{10}{1.54}\right) = 1$$

$$\text{LAN Utilization} = \left( \frac{\min(\text{avg data, Access link})}{1 \text{ Gbps}} \right) = \left( \frac{1.54 \text{ Mbps}}{1000 \text{ Mbps}} \right) = 1.54 \times 10^{-3}$$

بعد از 10 برابر کردن:

$$\text{Access Link Utilization} = \min\left(1, \frac{10}{15.4}\right) = 0.649 \sim 0.65$$

$$\text{LAN Utilization} = \left( \frac{\min(\text{avg data, Access link})}{1 \text{ Gbps}} \right) = \left( \frac{10 \text{ Mbps}}{1000 \text{ Mbps}} \right) = 0.01$$

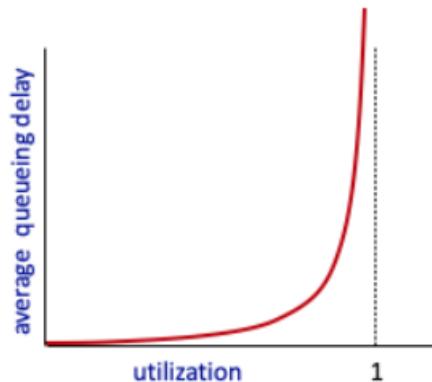
تغییرات با مقایسه این دو بخش قابل ملاحظه است.

(ب)

با استفاده از web cache مقدار Access Link Utilization در واقع در 60 درصد ضرب می شود که این مقدار همچنان از 1 بیشتر است. درنتیجه روش مناسبی نیست. پس 10 برابر کردن پهنانی باند بهتر عمل خواهد کرد که برای مقدار hit rate داریم:

$$(1 - x) \times \frac{10}{1.54} < \frac{10}{15.4} \Rightarrow x > 90\%$$

(ج)



می دانیم:

$$\text{delay} = 2 \text{ sec} + \text{access link delay} + \text{lan delay}$$

در حالت اول، با توجه به شکل بالا و همچنین مقدار utilization = 1، این زمان به بینهایت میل خواهد کرد.

در صورت استفاده از web cache هم نیز باز همین حالت رخ می دهد.

در حالت hit rate = 0.9 هم زمان در اردر میلی ثانیه (حدودا) خواهد بود.

با 10 برابر کردن پهنای باند نیز به 1 ثانیه (باز هم حدودا) خواهد رسید.

## سوال پنجم

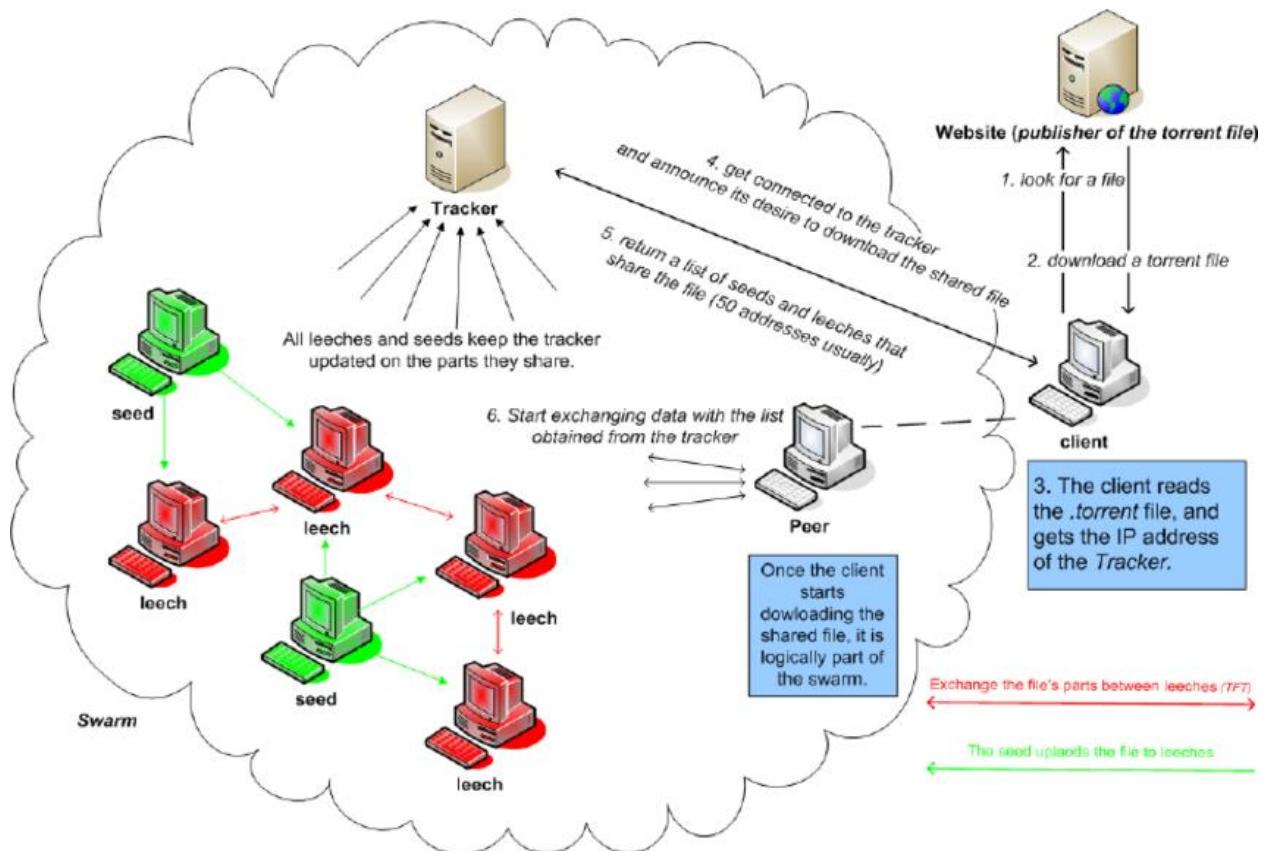
**سؤال ۵** در یک شبکه File Sharing مانند torrent کار می‌کند، توضیح دهید چگونه کاربری که تازه وارد شبکه شده می‌تواند بدون داشتن هیچ فایلی، فایل مورد نظر خود را دریافت کند؟ (راهنمایی: درباره‌ی optimistic unchoking تحقیق کنید).

### پاسخ سوال پنجم:

برای پاسخگویی به این سوال ابتدا مفهوم choking را معرفی می‌کنیم. سپس ویژگی optimistic unchoking را معرفی خواهیم کرد و تاثیر آن را نشان خواهیم داد.

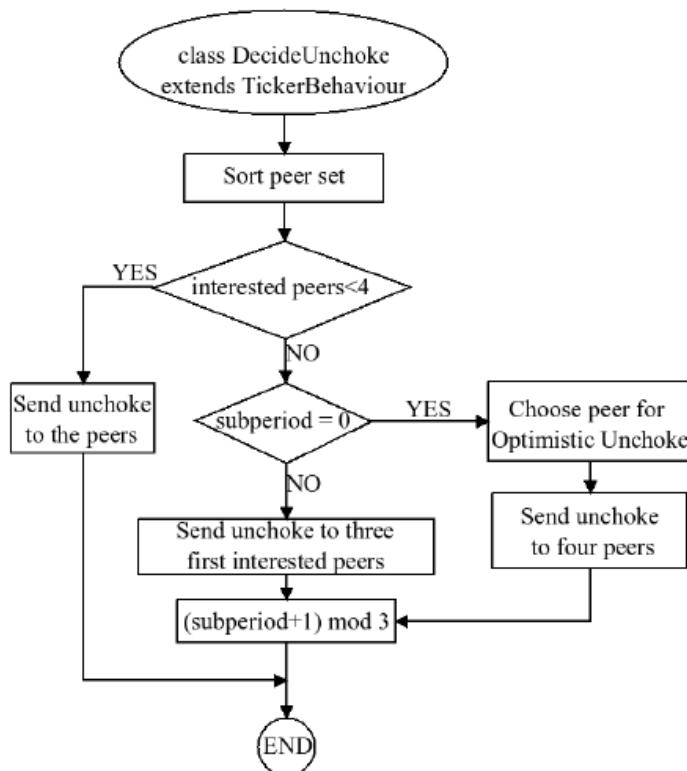
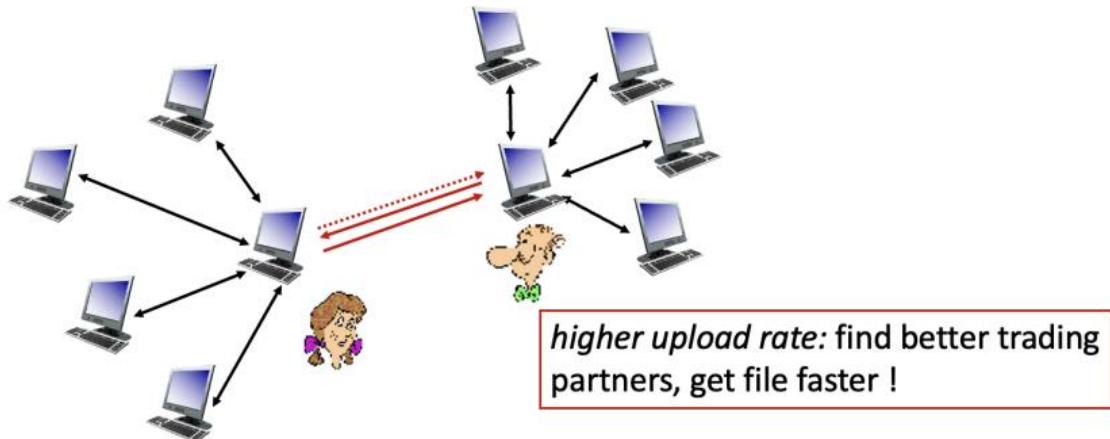
Choking: هنگامی که یک peer خاص وارد می‌شود، به فرآیندِ موقتِ خودداری کردن از آپلود قطعات فایل به آن peer مذکور، choking گفته می‌شود. از دلایل این امر می‌توان به avoid free riders و congestion اشاره نمود.

در شبکه‌های P2P مانند torrent، دریافت یک فایل بدون داشتن فایل از طریق unchoking می‌پذیرد. این مکانیزم در پروتوكل BitTorrent استفاده می‌شود. در بخش ابتدایی هنگام ورود یک peer، با peer‌های دیگر در شبکه ارتباط برقرار می‌کند. این peer درخواست خود برای دریافت قطعه‌های مختلف فایل را به peer‌های دیگر ارسال می‌نماید. این درخواست‌ها به تعداد محدودی از peer‌ها ارسال می‌شوند. برای این کار در ابتدا به tracker که حاوی لیست کاربرانی است که در حال به اشتراک گذاشتن آن فایل مورد نظر می‌باشند، دسترسی پیدا می‌کنیم. که هر کدام از این کاربران یک torrent هستند. توجه کنید که هنگام ارسال درخواست‌ها، لیست فایل‌هایی که نیاز دارد را اعلام می‌کند. حال ممکن است تمام peer‌ها، این کاربر که تازه وارد شده است را choke بنمایند و قطعه‌ای را برای او ارسال نکنند (چون کاربر جدید هیچ چیز به اشتراک نگذاشته است). در این بخش همان optimistic unchoking مطرح می‌شود. در اصل کلاینت‌های این پروتوكول از آن استفاده می‌کنند. هر کلاینت به طور متناوب peer را برمی‌گزیند، آن را unchoke کرده و تعدادی از قطعات خواسته شده را برای او می‌فرستد. حال که کاربر جدید قطعه دارد، می‌تواند در torrent شرکت نماید. زیرا می‌تواند قطعه‌هایی که دارد را برای دیگر peer‌ها ارسال نماید. حال که دیگر عضوی از شبکه است، می‌تواند فایل خود را دریافت کند و حتی تبدیل به یک seeder بشود. تصاویر پایین توضیحات بیشتر در مورد این فرآیند و پروتوكول می‌باشند.



# BitTorrent: tit-for-tat

- (1) Alice “optimistically unchoke” Bob
- (2) Alice becomes one of Bob’s top-four providers; Bob reciprocates
- (3) Bob becomes one of Alice’s top-four providers



## **Choking & Unchoking Strategy**

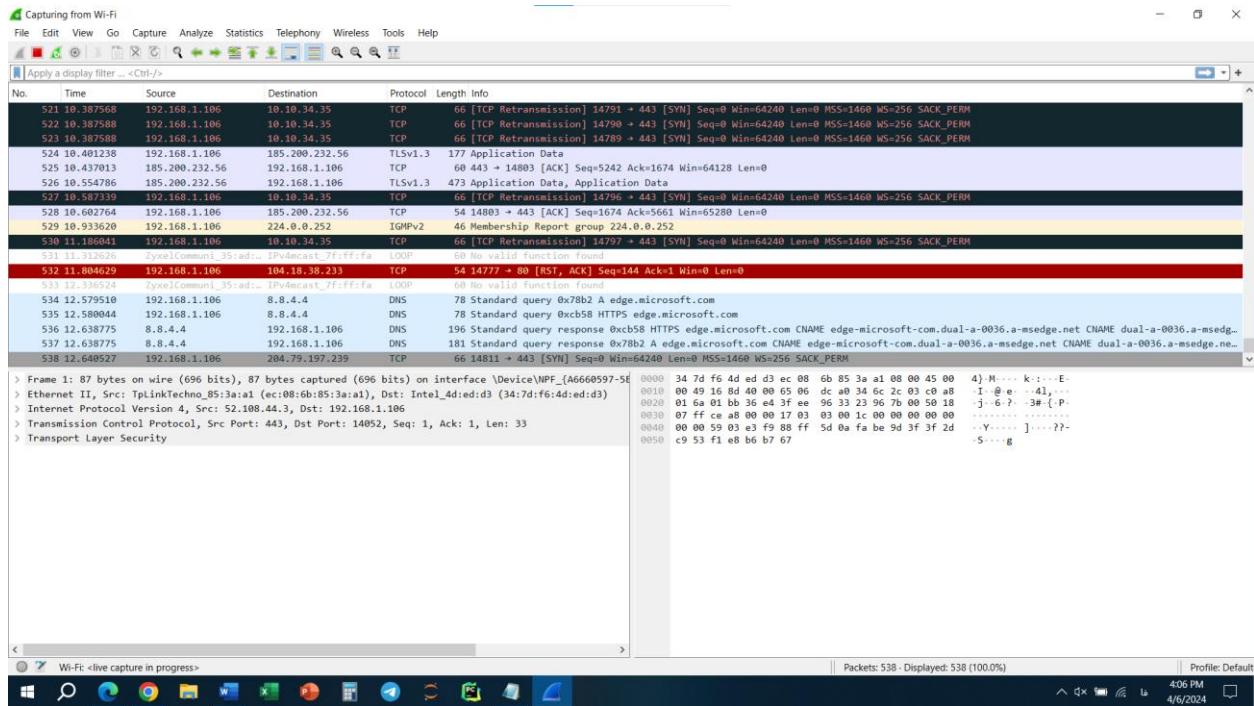
- All peer has limited connection capability. Each peer need to use tit-for-tatish strategies to ensure that they get a consistent download rate.
- Some criteria a good choking algorithm should meet:
  - Cap the number of simultaneous uploads for good TCP performance.
  - Avoid choking and unchoking quickly or 'fibrillation'.
  - Reciprocate to peers who let it download.
  - Try new peers once in a while to find better peers.
- By definition any peer may use any strategy.

## **Optimistic Unchoking Strategy**

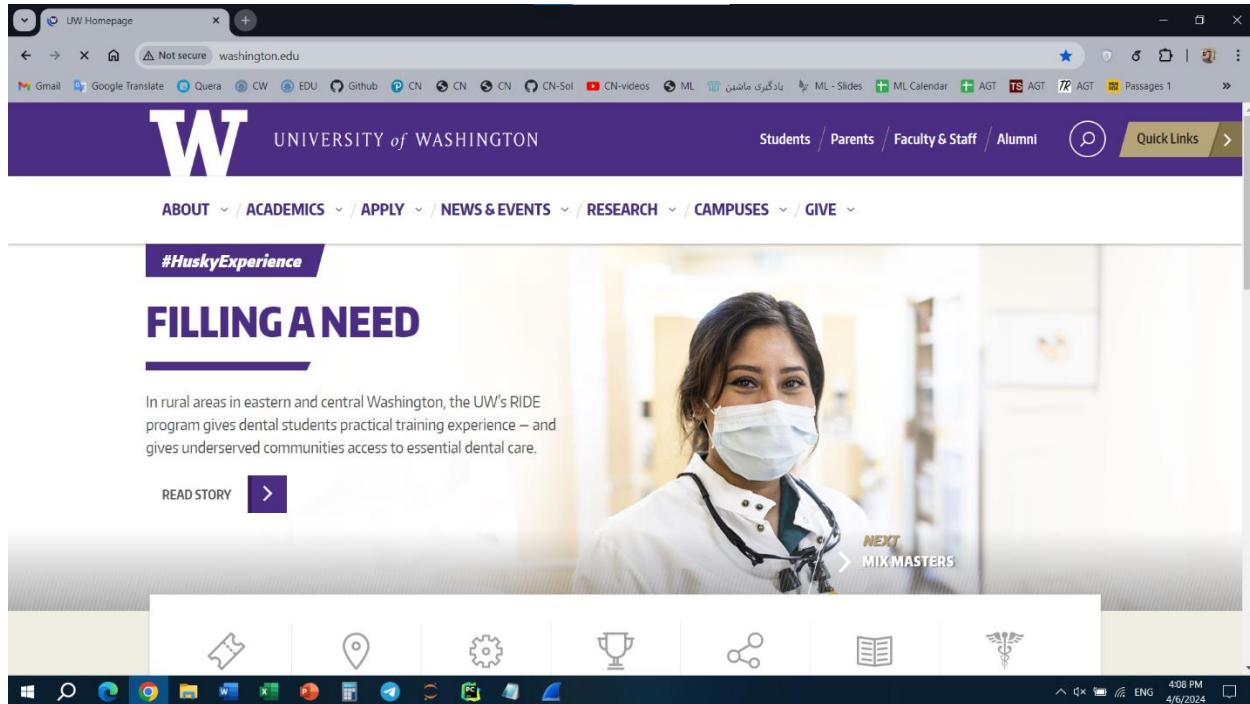
- The peer also keeps one connection available for searching better peers. This is called 'optimistic unchoking'.
- For optimistic unchoking, at any one time there is a single peer which is unchoked regardless of its upload rate (if interested, it counts as one of the four allowed *downloaders*).
- Which peer is optimistically unchoked rotates every 30 seconds.
- Newly connected peers are three times as likely to start as the current optimistic unchoke as anywhere else in the rotation. This gives them a decent chance of getting a complete piece to upload.

## سوال ششم

در این بخش به ترتیب عکس‌ها را قرار می‌دهیم و پاسخ هر بخش را مشخص می‌کنیم.



نرم‌افزار را اجرا و در حالت capture قرار داده‌ایم. در این بخش از سایت <http://www.washington.edu> کمک گرفته‌ایم.



حال نرمافزار را از حالت capture خارج کرده و بسته‌های http را فیلتر می‌نماییم.

Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

No.	Time	Source	Destination	Protocol	Length	Info
2902	58.371768	192.168.1.106	128.208.60.208	HTTP	487	GET / HTTP/1.1
3053	60.024001	128.208.60.208	192.168.1.106	HTTP	674	HTTP/1.1 200 OK (text/html)
3059	60.058714	192.168.1.106	128.208.60.208	HTTP	441	GET /static/home/wp-includes/css/dist/block-library/style.css?5d6af5a HTTP/1.1
3580	61.963226	128.208.60.208	192.168.1.106	HTTP	128	HTTP/1.1 200 OK (text/css)
3581	61.971186	192.168.1.106	128.208.60.208	HTTP	434	GET /static/home/wp-content/themes/uw-2014/style.css?ver=3.0.0 HTTP/1.1
3875	63.633719	128.208.60.208	192.168.1.106	HTTP	816	HTTP/1.1 200 OK (text/css)
3877	63.637262	192.168.1.106	128.208.60.208	HTTP	434	GET /static/home/wp-content/themes/boundless/style.css?75d6af5a HTTP/1.1
3927	64.589766	192.168.1.106	128.208.60.208	HTTP	414	GET /static/home/wp-includes/js/underscore.min.js?5d6af5a HTTP/1.1
3950	65.266469	128.208.60.208	192.168.1.106	HTTP	112	HTTP/1.1 200 OK (application/x-javascript)
3952	65.268908	192.168.1.106	128.208.60.208	HTTP	413	GET /static/home/wp-includes/js/jquery/jquery.js?5d6af5a HTTP/1.1
4299	66.672208	192.168.1.106	34.104.35.123	HTTP	453	HEAD /edged1/difffgen-puffin/fnfpkpm1hhgieaddgfemjhofmblmnib/1.71e653429ce3b7e8800b6674a683e283896ad5df5e58ec5c38b1b2f7350719b74...
4689	67.475784	128.208.60.208	192.168.1.106	HTTP	60	HTTP/1.1 200 OK (application/x-javascript)
4612	67.477785	192.168.1.106	128.208.60.208	HTTP	421	GET /static/home/wp-includes/js/jquery-migrate.js?5d6af5a HTTP/1.1
4698	67.742196	128.208.60.208	192.168.1.106	HTTP	1414	[TCP Previous segment not captured] Continuation
4701	67.743692	128.208.60.208	192.168.1.106	HTTP	1414	[TCP Previous segment not captured] Continuation
4708	67.746615	128.208.60.208	192.168.1.106	HTTP	1414	Continuation
4709	67.747809	128.208.60.208	192.168.1.106	HTTP	1414	Continuation
4711	67.748846	128.208.60.208	192.168.1.106	HTTP	1414	[TCP Previous segment not captured] Continuation

```

> Frame 4: 197 bytes on wire (1576 bits), 197 bytes captured (1576 bits) on interface \Device\NPF_{A666055
> Ethernet II, Src: Intel Dual Band (34:7d:f6:4d:ed:d3), Dst: TP-Link Techno_85:3a:a1 (ec:08:0b:85:3a:a1)
> Internet Protocol Version 4, Src: 192.168.1.106, Dst: 104.18.38.233
> Transmission Control Protocol, Src Port: 14777, Dst Port: 80, Seq#: 1, Ack#: 1, Len: 143
> Hypertext Transfer Protocol

```

Packets: 10210 - Displayed: 114 (1.1%) - Dropped: 0 (0.0%)

Profile: Default

.1

همانطور که مشخص است هر دو نسخه HTTP/1.1 را اجرا می‌نمایند.

2902 58.371768	192.168.1.106	128.208.60.208	HTTP	487 GET / HTTP/1.1
3053 60.024001	128.208.60.208	192.168.1.106	HTTP	674 HTTP/1.1 200 OK (text/html)

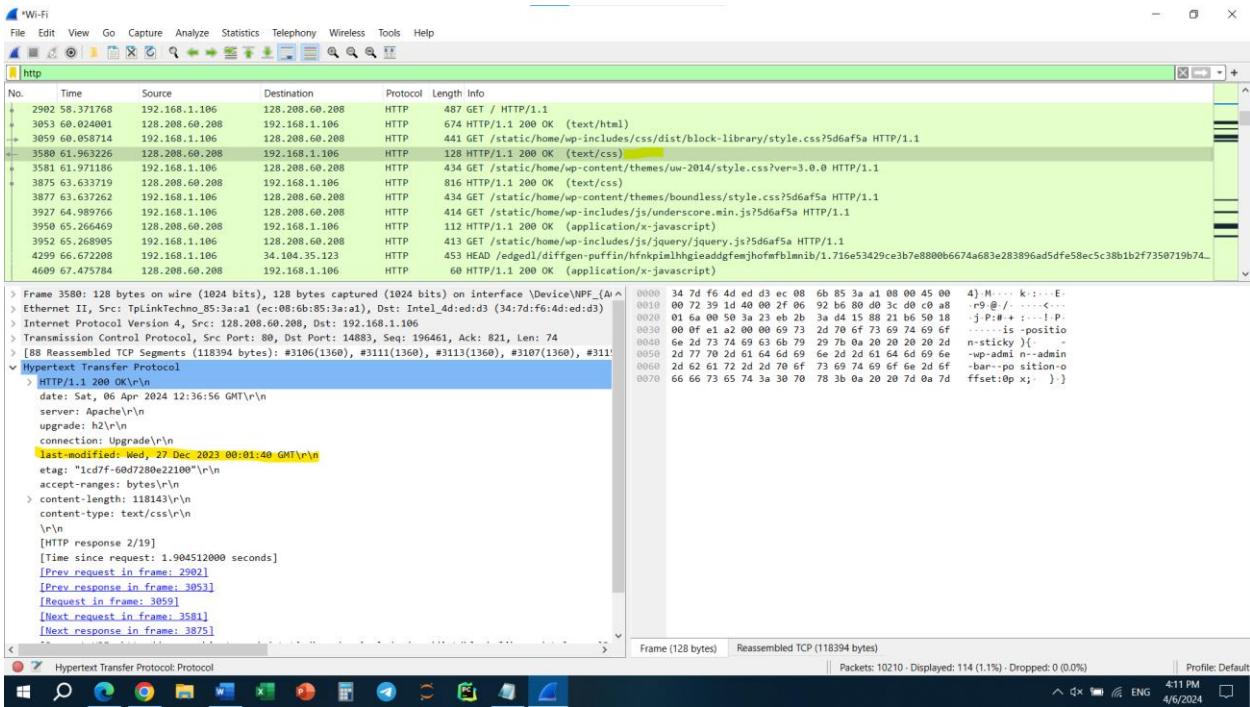
.2

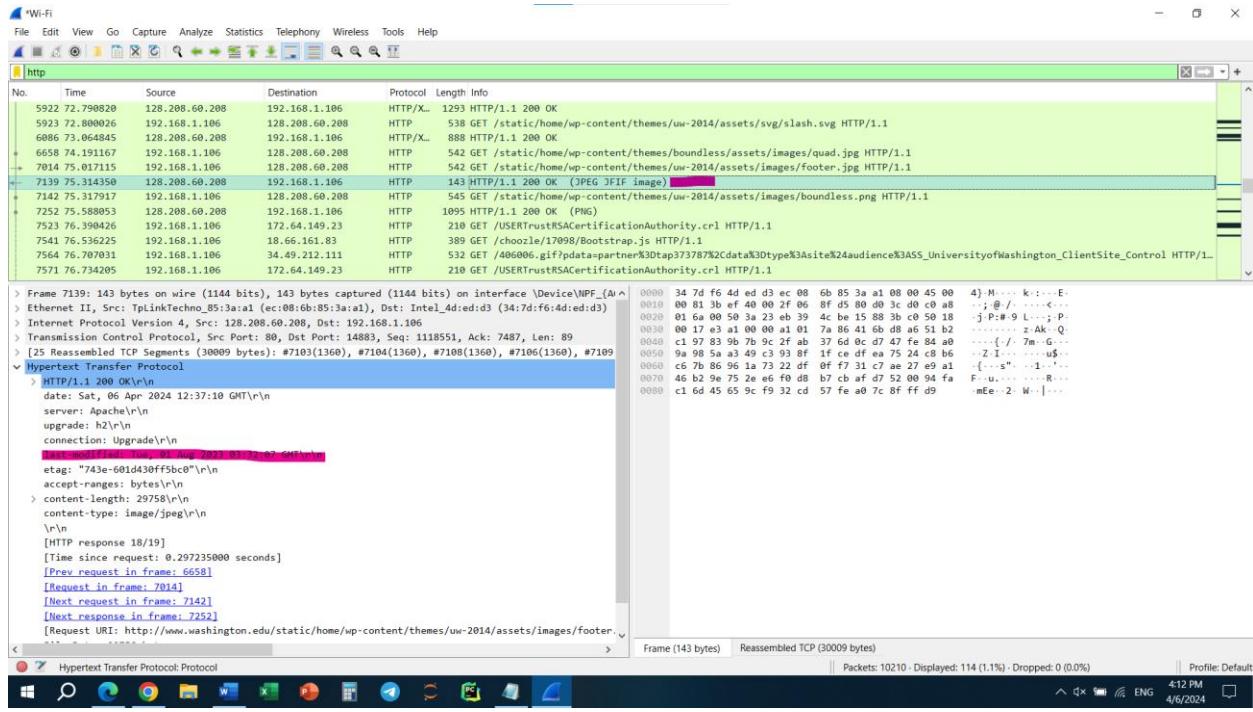
آدرس‌ها در شکل زیر مشخص شده‌اند. آدرس سبز رنگ برای سرور و آدرس نارنجی رنگ برای ما می‌باشد.

2902 58.371768	192.168.1.106	128.208.60.208	HTTP	487 GET / HTTP/1.1
3053 60.024001	128.208.60.208	192.168.1.106	HTTP	674 HTTP/1.1 200 OK (text/html)

.3

آخرین زمان تغییر 2تا از فایل‌ها در دو عکس زیر قابل ملاحظه است.

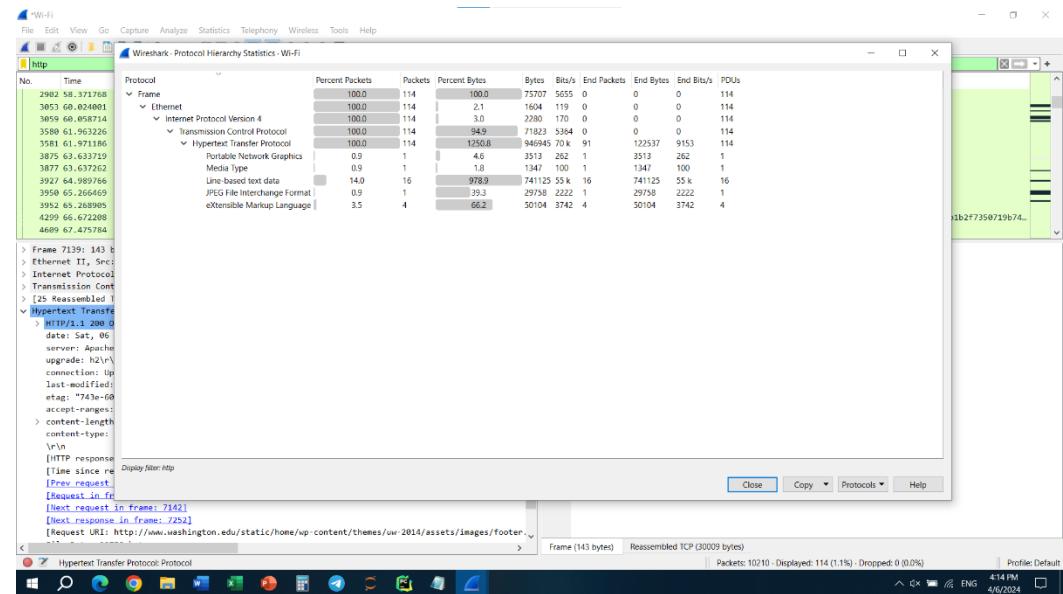




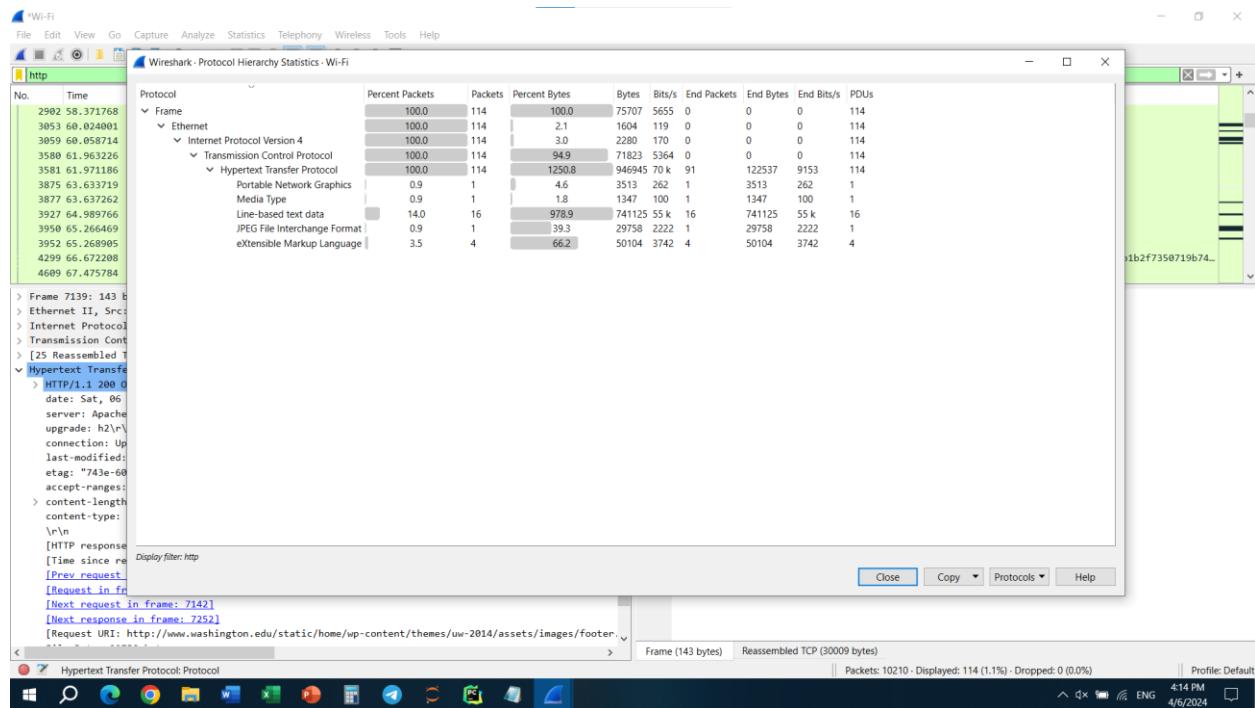
از last-modified این اطلاعات را پیدا کرده‌ایم.

.4

با استفاده از بخش conversations هم میتوانستیم استفاده نماییم. از بخش protocol hierarchy و statistics



تصویر بزرگتر:



5. به دلیل اینکه سایت شلوغی را انتخاب کردیم، فایل‌های بیشتری نیز دریافت نمودیم که چندتای آنان را در تصاویر زیر نشان می‌دهیم.

```

[Request in frame: 9370]
[Request URI: http://munchkin.marketo.net/163/munchkin.js]
Content-encoded entity body (gzip): 4741 bytes -> 11133 bytes
File Data: 11133 bytes
Line-based text data: application/x-javascript (26 lines)
/*\n * Copyright (c) 2007-2023, Marketo, Inc. All rights reserved.\n * See https://developers.marketo.com/MunchkinLicense.pdf for license terms\n * Marketo marketing automation web activity tracking script\n * Version: 163 r896\n*/
/*\n[truncated] (function(l){if(!l.MunchkinTracker){var h=l.document,p=h.location,C=encodeURIComponent[truncated],b,f=function(b,a){return b.className.match(RegExp("(\\s|^)+"+"(\\w|\\$|")"+"),)=e(l.XMLH[truncated],j=[s[1]]=s(n));return c},H=function(b){try{var a=b.createElement("a");a.href=b;return G[truncated]4==g&&a.test(b)?f=4:2==d[g-1].length&&1<g&&"co"==d[g-2]&&(f=3)):else if(2<d[g-1].le[truncated]g);if("!=a"for(a=a.split(";"),c=0;c<a.length;c+=1)if(d=a[c].replace(/^\s+|\s+$|g,""))[truncated]._itmMitigationForAll:[1],k=null,m=null,w=1.navigator.cookieEnabled||h.hasOwnProperty("c[truncated].window.navigator.sendBeacon(b):(c=(new Date).getTime(),e!=&&f.asyncOnly,0==a,indexOf[truncated].length;a+=1"hidden"==b[a].type&&(b[a].value=c),x=function(b,a,c,d){var g=null,h:h[truncated]f.wsInfo),"clickLink"==b&&(a._mchCn=e(f.customName),"",a._mchHo=h,a._mcl[truncated],n,a,c,d))else A.push(["post",arguments]),R=function(b){var a=b||1.event,c=a.target||[truncated]"#")&&!-=c.indexOf("javascript")&&F(a,"mchNoDecorate")?a:null};e(a)&&!Y(b,a)&&(d._mcl[truncated]"/",a,d),d&&(c=~("_mktotrkr")),id||c(c.id))return m=c,Q(),c||else return null)jelse A.push[truncated]jt"+"/getCookie?_mchid="+k+"&_mchid="+q,[credentials:"include"])),then(function(b){window.

```

Frame (1218 bytes) | Reassembled TCP (5244 bytes) | Uncompressed entity body (11133 bytes) | Packets: 10210 · Displayed: 114 (1.1%) · Dropped: 0 (0.0%) | Profile: Default

4:17 PM 4/6/2024

6

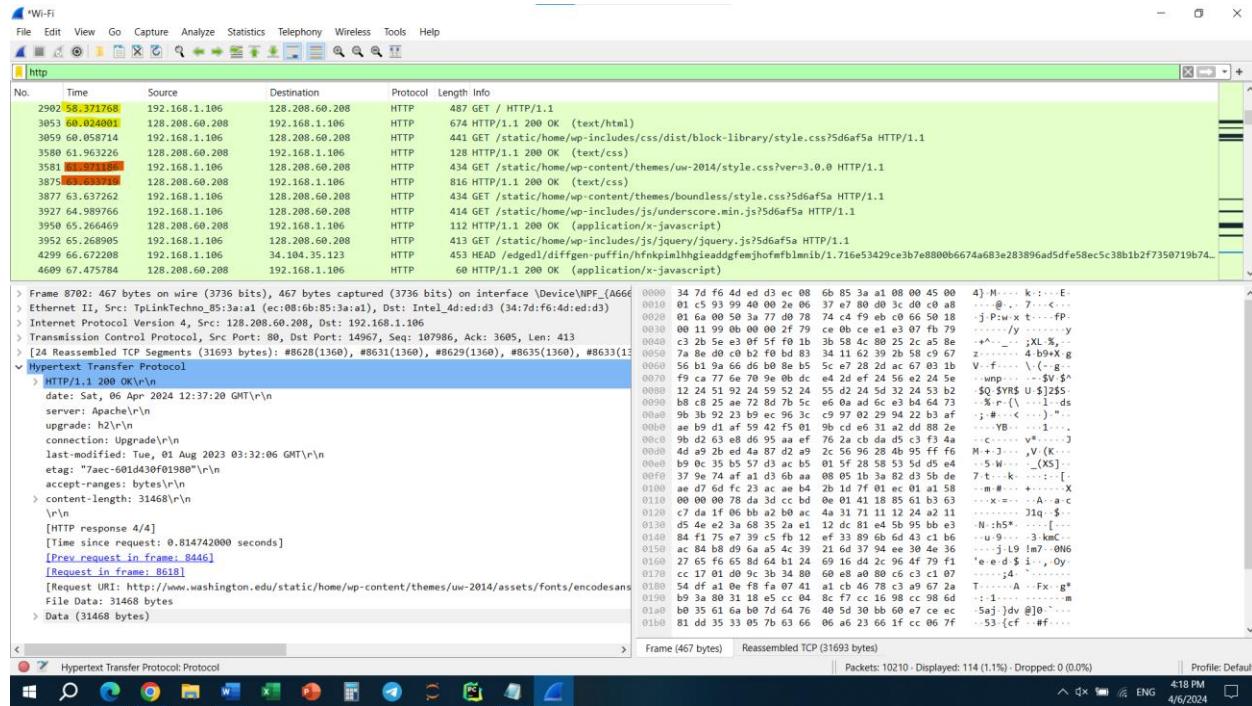
برای تفاوت زمان میان ارسال درخواست و دریافت پاسخ، در تصویر زیر برای 2 مرحله این تفاوت نشان داده شده است. این زمان با توجه به سنگینی سایت و محتوای آن، معقول به نظر می‌رسد.

برای مرحله اول:

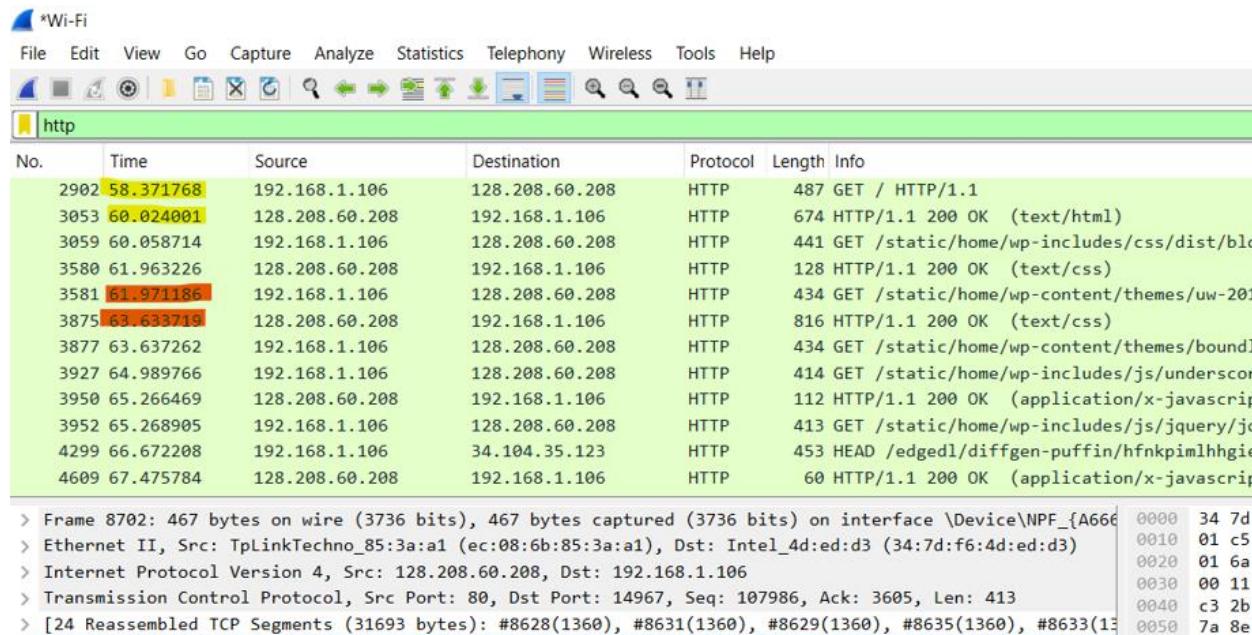
$$60.02 - 58.37 = 1.65$$

برای مرحله دوم:

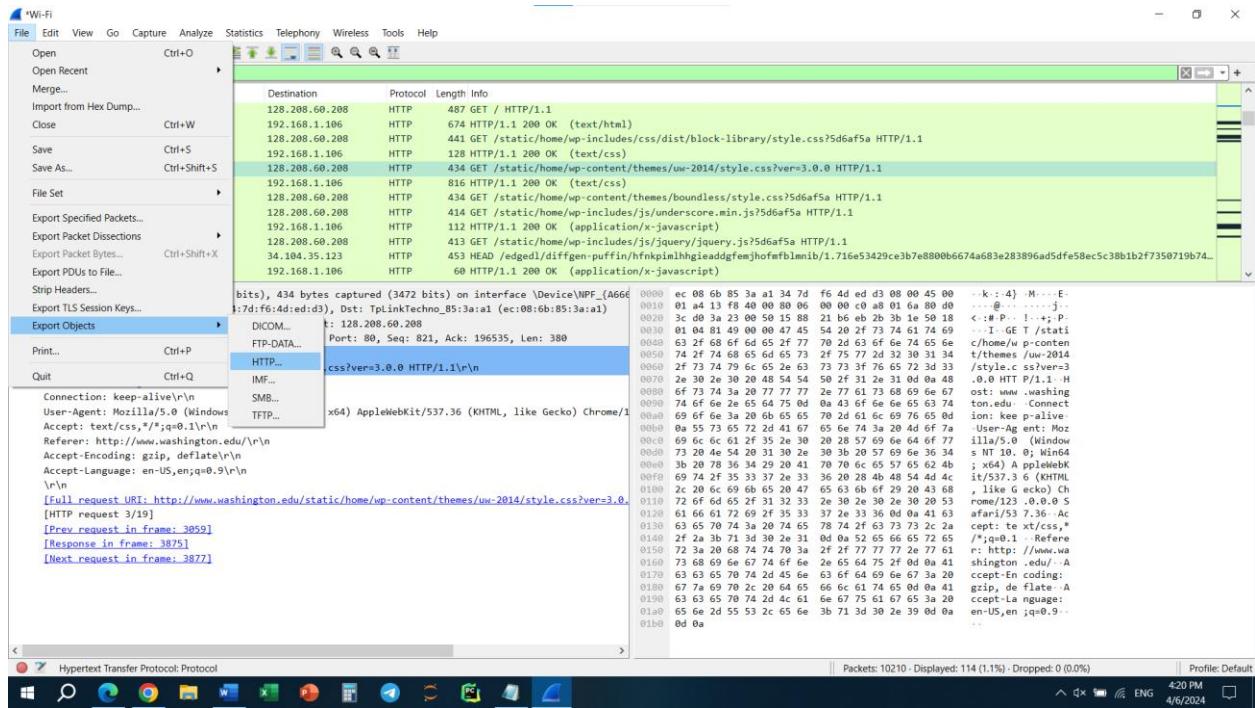
$$63.63 - 61.97 = 1.66$$



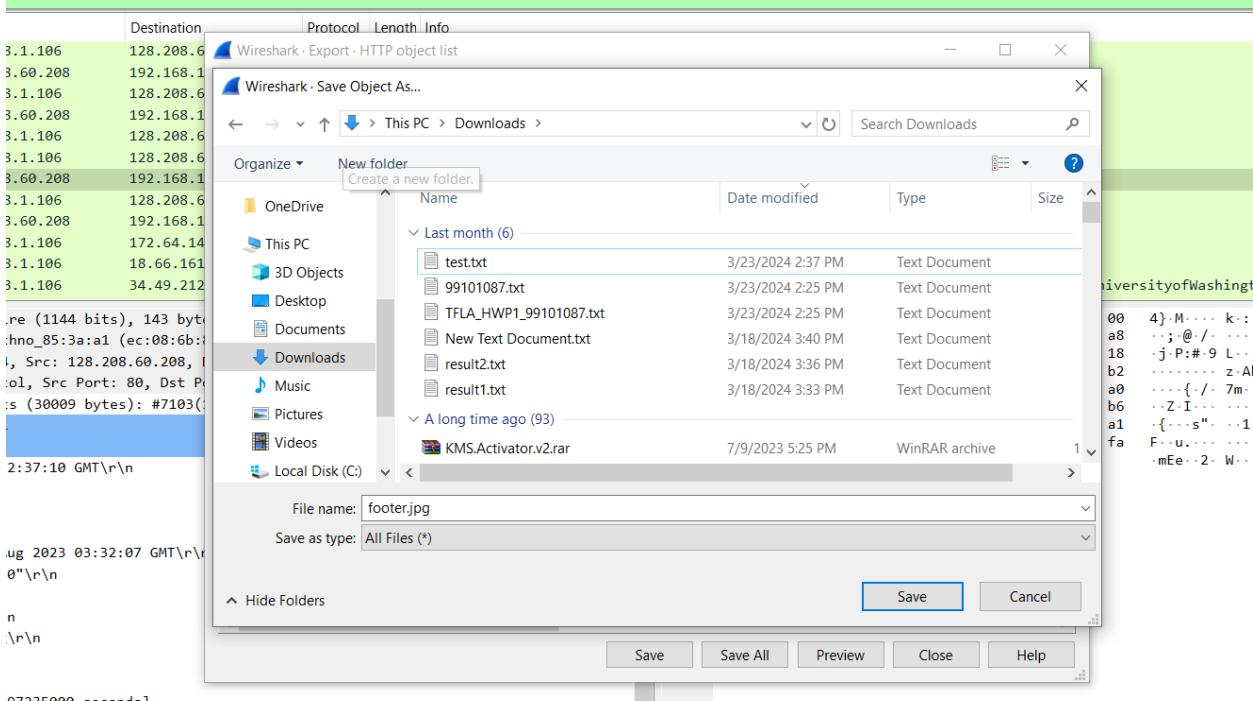
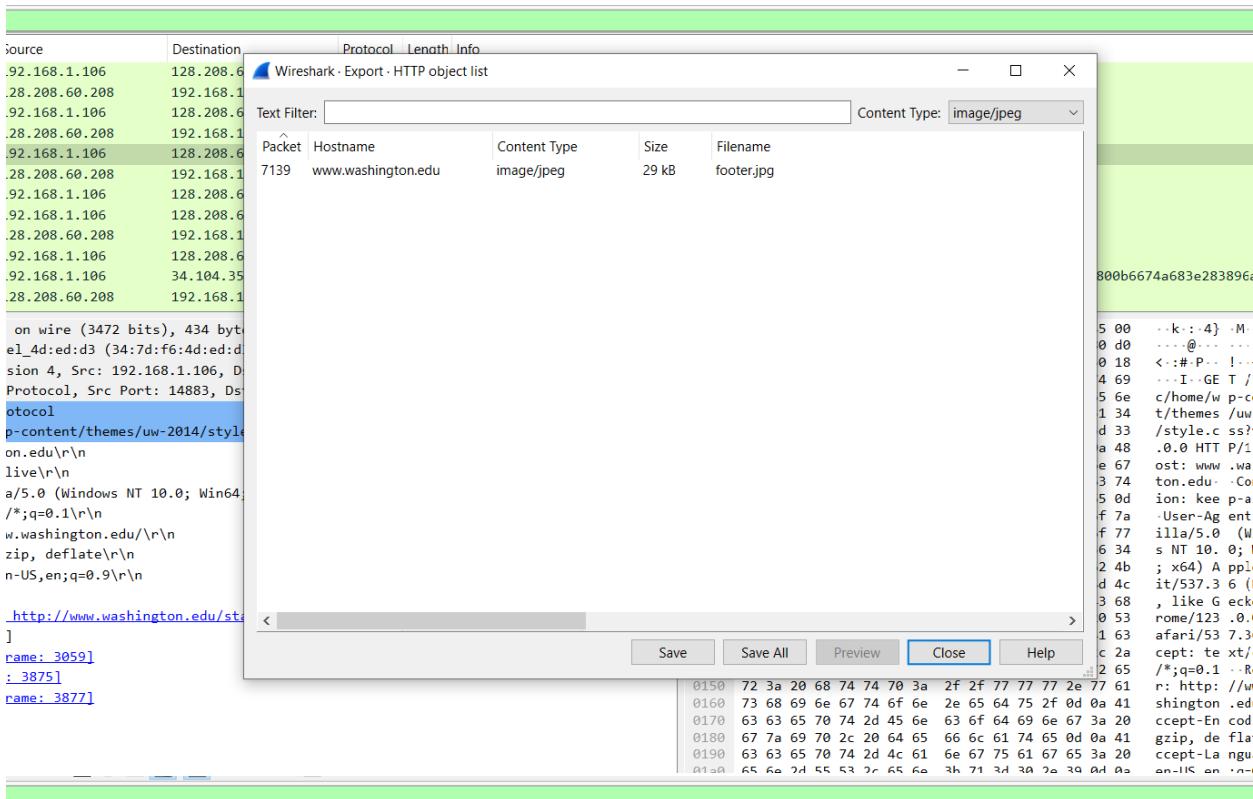
عكس واضح تر:



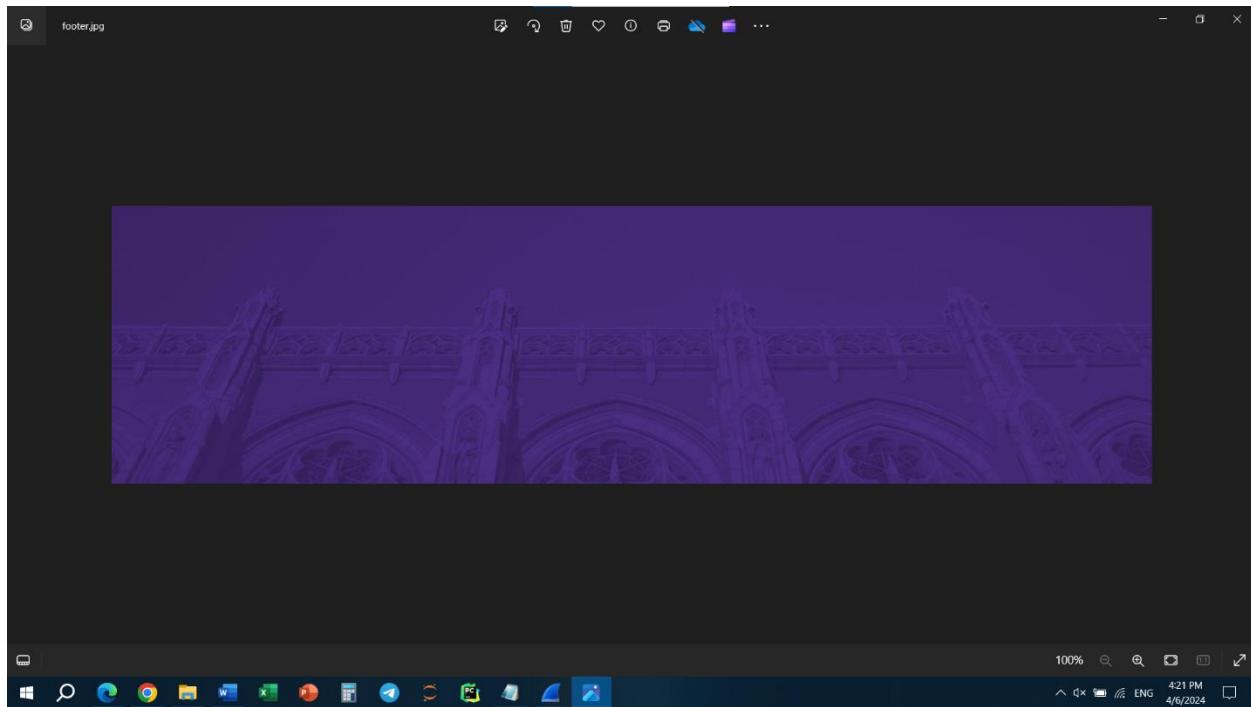
برای بازیابی تصویر، از بخش export objects، قسمت HTTP را انتخاب می‌کنیم.



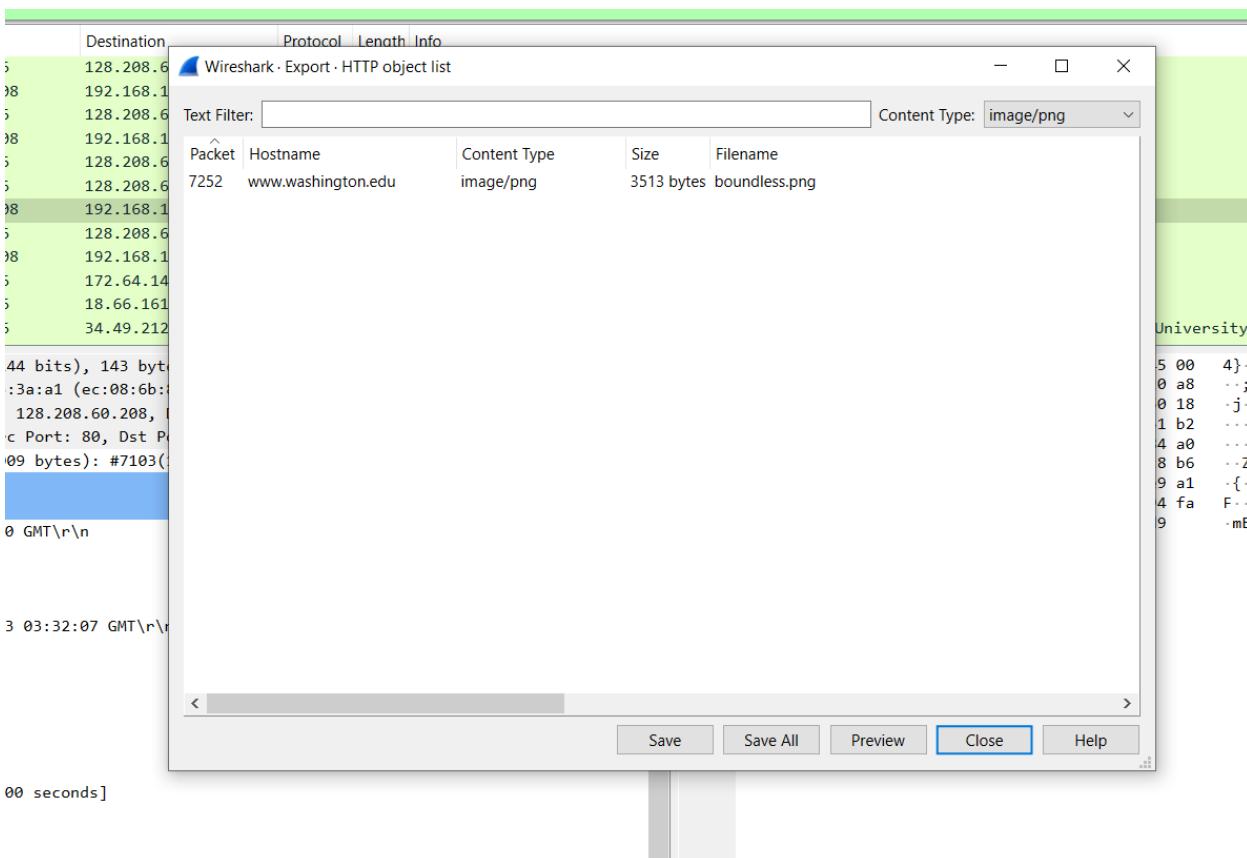
سپس می‌توانیم تصاویر را در این بخش مشاهده و سیو کنیم. برای مثال 2 تا از عکس‌های دریافت شده را نشان می‌دهیم.



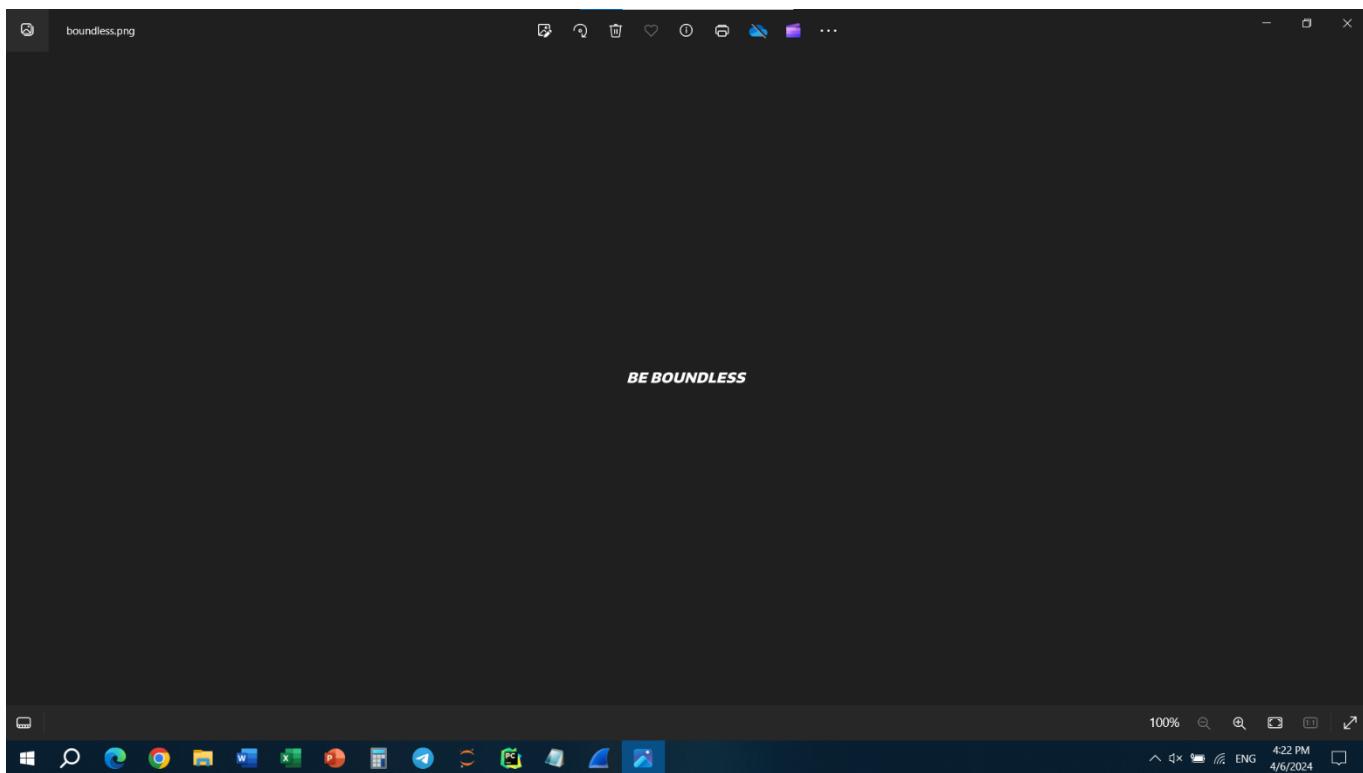
حال بعد ذخیره تصاویر داریم:



یک تصویر دیگر را هم نشان می‌دهیم:



که به شکل زیر است:



حال این دو تصویر را در سایت اصلی نشان می‌دهیم:

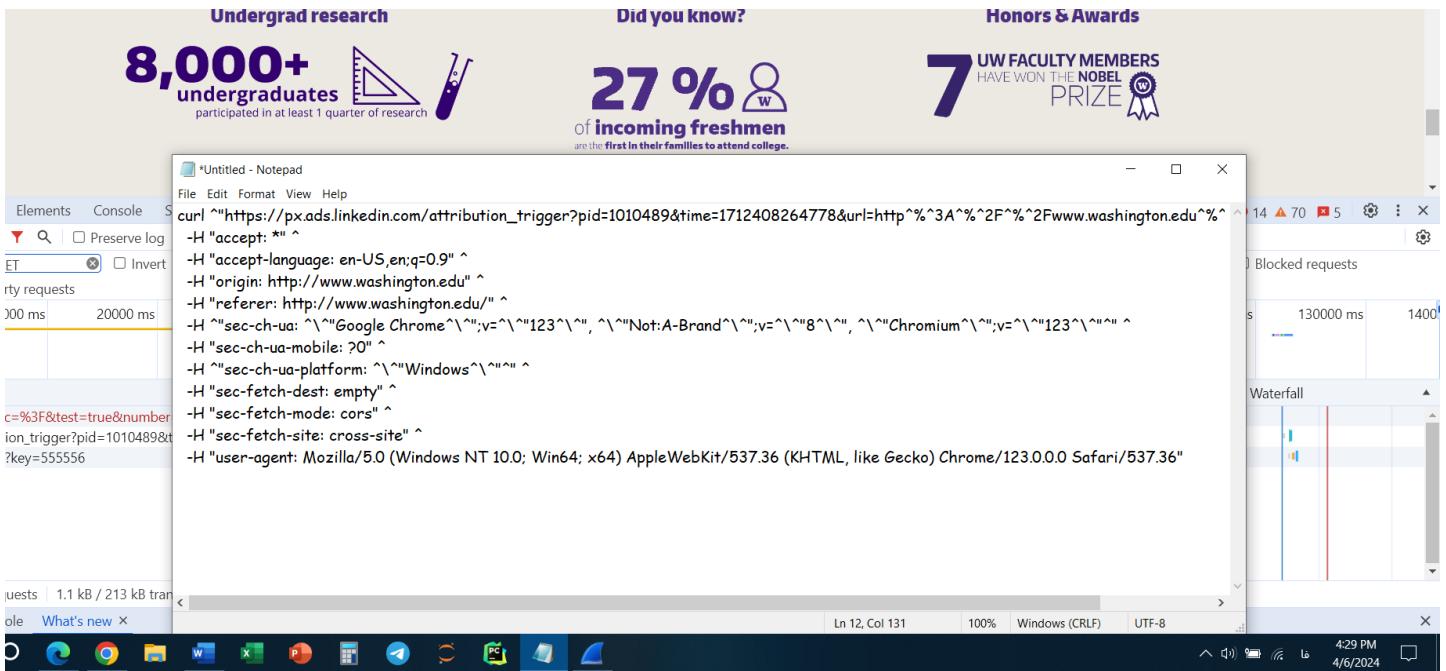


.8

برای این قسمت سوال، به بخش گفته شده می‌رویم و دستورهای GET را فیلتر می‌کنیم و uCRL یکی از آن‌ها را کپی می‌کنیم.

A screenshot of the NetworkMiner tool interface. The main window displays a HAR (HTTP Archive) file. The "Headers" tab is selected, showing requests for "/posts/?c=%3F&amp;test=true", "attribution\_trigger?pid=10100000000000000000000000000000", and "IsGDPR?key=555556". The "Content" tab shows the raw content of the first request. A context menu is open over the first request, with the "Copy" submenu expanded. Sub-options include "Copy URL", "Copy as cURL (cmd)", "Copy as cURL (bash)", "Copy as PowerShell", "Copy as fetch", "Copy as fetch (Node.js)", "Copy response", "Copy stack trace", "Copy all URLs", and "Copy all as HAR". The background shows the University of Washington homepage with sections like "FAST FACTS", "Did you know?", "Honors &amp; Awards", and a "Performance insights" waterfall chart.

که برابر زیر است:



که دستور آن به طور زیر است:

```
curl
^"https://px.ads.linkedin.com/attribution_trigger?pid=1010489&time=1712408264778&url=http%3A%2F%2Fwww.washington.edu%2F&tm=gtmv2%" ^
-H "accept: *" ^
-H "accept-language: en-US,en;q=0.9" ^
-H "origin: http://www.washington.edu" ^
-H "referer: http://www.washington.edu/" ^
-H ^"sec-ch-ua: \\"Google Chrome\\\";v=\\"123\\\", \\"Not:A-Brand\\\";v=\\"8\\\", \\"Chromium\\\";v=\\"123\\\"\\\"^"
-H "sec-ch-ua-mobile: ?0" ^
-H ^"sec-ch-ua-platform: \\"Windows\\\"^"
-H "sec-fetch-dest: empty" ^
-H "sec-fetch-mode: cors" ^
-H "sec-fetch-site: cross-site" ^
-H "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"
```

```
-H ^"sec-ch-ua-platform: ^\^"Windows^\^"^\^"  
-H "sec-fetch-dest: empty" ^  
-H "sec-fetch-mode: cors" ^  
-H "sec-fetch-site: cross-site" ^  
-H "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36  
(KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"
```

حال به طور مختصر نقش 3 مورد از header ها را توضیح می‌دهیم.

### 1. -H "accept-language: en-US,en;q=0.9"

این header زبان‌هایی که کلاینت ترجیح می‌دهد به آنها پاسخ را دریافت کند مشخص شده است. به نوعی وزن و ترجیحات آن زبان برای کلاینت می‌باشد. برای مثال دستور بالا نشان می‌دهد که زبان English یا همان American English را هم می‌پذیرد.

### 2. -H "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)

AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"

این header، نرمافزار کلاینت را مشخص می‌کند. برای مثال در بخش بالا، می‌گوید درخواست از سمت یک chrome با ورژن 123.0.0.0 است و از سیستم عامل Windows NT 10.0; Win64; x64 بهره می‌گیرد.

## Chrome UA string

The Chrome (or Chromium/Blink-based engines) user agent string is similar to Firefox's. For compatibility, it adds strings like KHTML, like Gecko and Safari.

## Examples

```
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/51.0.2704.103 Safari/537.36
```

3. -H "origin: <http://www.washington.edu>"

این `header`, منشا (scheme, hostname, port) که باعث درخواست شده است را نشان می‌دهد.

## Origin

The `Origin` request header indicates the [origin](#) (scheme, hostname, and port) that caused the request. For example, if a user agent needs to request resources included in a page, or fetched by scripts that it executes, then the origin of the page may be included in the request.

4. -H "sec-fetch-mode: cors"

این `header`, در واقع `mode` درخواست را نشان می‌دهد. که در بخش بالا به این معناست که درخواست یک پروتکل `cors` است.

## Sec-Fetch-Mode

The `Sec-Fetch-Mode` [fetch metadata request header](#) indicates the `mode` of the request.

`cors`

The request is a [CORS protocol](#) request.