CS-374/672 Computer Networks: Fall 2015

Assignment 2

In this assignment, we will look at various HTTP tools to understand page load dynamics, and develop heuristics to improve the perceived user performance.

1. Start with experimenting with the following tools

    a. Firebug extension to Firefox, to log details of all HTTP requests made to download a particular webpage https://addons.mozilla.org/en-US/firefox/addon/firebug/

    b. Net:Export extension to Firebug, to save HTTP logs in HAR format https://getfirebug.com/releases/netexport/

    c. HAR viewer to study HAR files http://www.softwareishard.com/har/viewer/

    d. Understand the HAR JSON format, which you will need to parse the HAR files http://www.softwareishard.com/blog/har-12-spec/

    e. Tshark, which comes bundled with Wireshark https://www.wireshark.org/docs/man-pages/tshark.html

    You will need this because the HAR file produced by Firebug/Net:Export does not carry a unique connection identifier which you will need for some parts of the assignment, so you will need to find the connections using the .pcap files captured by wireshark. A command like the one below will give the (IP-src, IP-dst, Port-src, Port-dst) identifiers for the different TCP connections and objects downloaded on these connections:

    ```
    tshark -r vox.pcap -Y "ip.dst == 103.245.222.143 && http" -T fields -e ip.src -e ip.dst
    -e tcp.port -e col.Info
    ```

2. Webpages to probe

    a. Remember to clear your cache and DNS before starting. Try to not use the IIT network, and instead use the .pcap and .har files provided by me, or download new files using a 3G connection

    b. We will study two webpages

        i. http://www.nytimes.com/ which has a lot of small objects

        ii. http://www.vox.com/a/maps-explain-the-middle-east which too has several objects, but some quite large in size

c. Note that the second website is responsive and images displayed lower down the page are not downloaded right away. So remember to scroll down the page as soon as you type the URL

d. Save the .har and .pcap dumps for each website for further analysis.

3. Analysis

a. Write scripts to parse the HAR JSON, for each website, and find out the following:

i. Total number of objects downloaded, and number of objects downloaded from each domain

ii. Total size of objects downloaded, and size of content downloaded from each domain

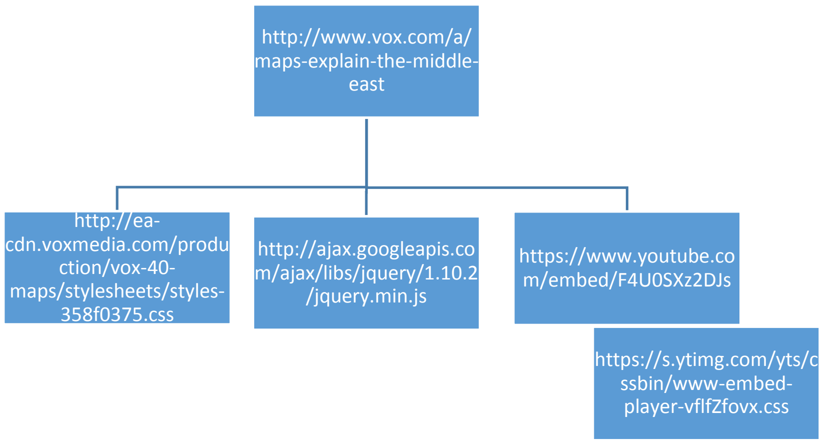
iii. Different types of objects downloaded, ie. images, html, javascript, css, etc. Use the filename extension of the objects to classify them

iv. Number of TCP connections opened to each domain, and number and size of objects downloaded on each connection

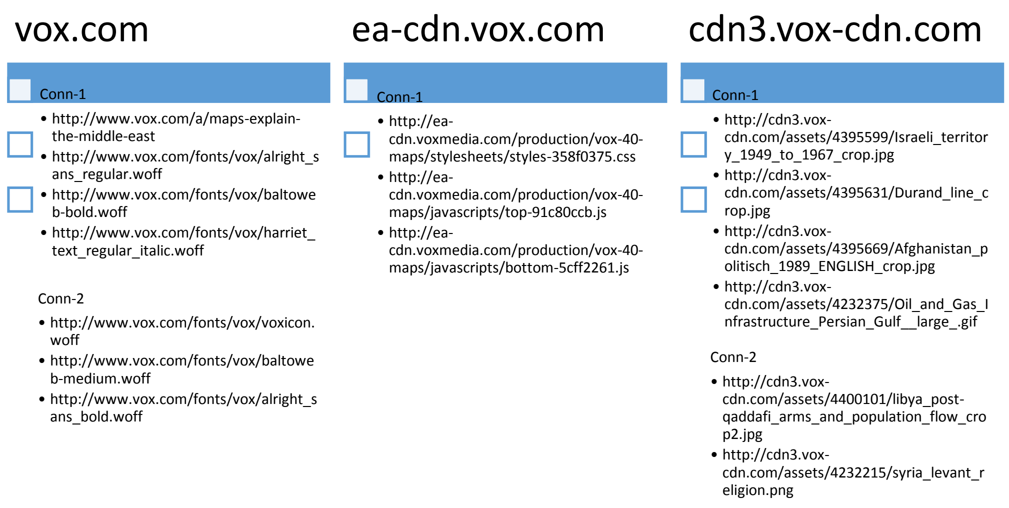
v. Your script should take the HAR file as an argument and produce the above output in a nice readable format

b. Augment your scripts that as you parse the HAR file, you construct an object tree and download tree as follows:

i. Use the referrer header in each object to build the download tree, which will look something as follows. The root page will be the referrer for objects referred by it, and additional objects such as CSS files and JS may further be referrers for objects referred by them. The tree is likely to be quite flat though, with most objects appearing just one level below the root.


http://www.vox.com/a/maps-explain-the-middle-east
http://ea-cdn.voxmedia.com/production/vox-40-maps/stylesheets/styles-358f0375.css
http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js
https://www.youtube.com/embed/F4U0SXz2DJs
https://s.ytimg.com/yts/cssbin/www-embed-player-vflfZfovx.css

ii. Use the connection identifier information in the HAR JSON to create a download tree as follows. List the domains from which objects are downloaded, TCP connections for each domain, and objects downloaded on each TCP connection.



vox.com

Conn-1

- http://www.vox.com/a/maps-explain-the-middle-east
- http://www.vox.com/fonts/vox/alright_sans_regular.woff
- http://www.vox.com/fonts/vox/baltoweb-bold.woff
- http://www.vox.com/fonts/vox/harriet_text_regular_italic.woff

Conn-2

- http://www.vox.com/fonts/vox/voxicon.woff
- http://www.vox.com/fonts/vox/baltoweb-medium.woff
- http://www.vox.com/fonts/vox/alright_sans_bold.woff

ea-cdn.vox.com

Conn-1

- http://ea-cdn.voxmedia.com/production/vox-40-maps/stylesheets/styles-358f0375.css
- http://ea-cdn.voxmedia.com/production/vox-40-maps/javascripts/top-91c80ccb.js
- http://ea-cdn.voxmedia.com/production/vox-40-maps/javascripts/bottom-5cff2261.js

cdn3.vox-cdn.com

Conn-1

- http://cdn3.vox-cdn.com/assets/4395599/Israeli_territory_1949_to_1967_crop.jpg
- http://cdn3.vox-cdn.com/assets/4395631/Durand_line_crop.jpg
- http://cdn3.vox-cdn.com/assets/4395669/Afghanistan_politisch_1989_ENGLISH_crop.jpg
- http://cdn3.vox-cdn.com/assets/4232375/Oil_and_Gas_Infrastructure_Persian_Gulf__large_.gif

Conn-2

- http://cdn3.vox-cdn.com/assets/4400101/libya_post-qaddafi_arms_and_population_flow_crop2.jpg
- http://cdn3.vox-cdn.com/assets/4232215/syria_levant_religion.png

iii. Your script should output this data in two text files in a standard structure, for example:

1. Object tree: A list of [node id, URL, parent node id] tuples, where the node ids can be simple auto increment integers to uniquely identify a URL

2. Download tree: A list of [domain name, connection id, URL] tuples, where the connection id can be simple auto increment integers to identify a unique TCP stream

c. Timing analysis. Use the data structures compiled above to study the following.

i. Find the page load time, ie. time from sending the first DNS request to the end time of loading the last object

ii. For each domain, find the time spent in the DNS query when opening the first TCP connection. Also confirm that for subsequent TCP connections for each domain, a fresh DNS query was not launched, ie. the IP address was picked up from the DNS cache. If you notice any strange results like DNS latency being reported as zero always, you must explain these observations. It is not enough to just give the results, you will be evaluated equally on how well you understand the results.

iii. For each TCP connection in the download tree, use the timing information in the HAR file to find the following:

1. Connection establishment time to open the TCP connection, reported in the *connect* variable in the HAR

2. Total time spent in waiting for the server to respond, reported in the *wait* variable in the HAR

3. Total time spent in receiving data, reported in the *receive* variable in the HAR

4. Total time spent in sending data, reported in the *send* variable in the HAR. This should ideally be 0

5. Note that on a per-connection basis, you can sum the total times for waiting/sending/receiving reported for each object, since the object downloads will occur serially. However, you cannot sum these times across connections because the downloads may be happening simultaneously

6. Total time for which the connection was active, calculated from the time when the first request was sent, to the time when the last request was received

7. Now, you can use the above data to find the idle and active periods of the connection:

   active percentage = [sending + waiting + receiving] / [total active time]

   idle percentage = 1 – active

8. Find the average achieved goodput of the connection

   average goodput = [total data received] / [total time spent in receiving]

9. Find the maximum achieved goodput of the connection, by identifying the largest object downloaded on the connection, and then calculating the goodput

   maximum goodput = [largest object size] / [receive time of object]

   For greater accuracy, also conduct a separate experiment by downloading these largest objects through a direct request rather than via the webpage, and find the maximum goodput. Does this differ from the average achieved goodput you calculated in the previous part? Why?

10. Also find the average achieved goodput of the network, ie. across all domains

11. Do you think your download capacity was utilized well to access the web pages? Hint: Compare the average achieved goodput of the network, with the maximum of the maximum achieved goodput on all connections.

iv. Browser scheduling policies

1. Run through the starting and ending times of each TCP connection and find the maximum number of TCP connections opened per domain.

2. Similarly, find the maximum number of TCP connections across domains that are simultaneously active.

        3. Do you see any patterns, such as a cap imposed by the browser on the number of TCP connections opened per domain, or the total number of TCP connections, or the number of objects being downloaded per TCP connection?

    v. All the data in part (c) above should be output by your scripts in a neat readable manner

d. What-if scenarios. We want to see how we could possibly improve on the total page load time

    i. As a first level of optimization, assume that you can collapse on each TCP connection all the GET requests on that connection, so that the objects are received back to back and there are no idle times between requests and waiting times for requests. This indicates a browser scheduling policy with no caps on the number of objects to be downloaded simultaneously per connection. How would you compute the expected page load time in this case? Be careful to consult the object tree when collapsing the requests for objects – you do not want to download an object until its parent has been downloaded entirely.

    ii. Run another calculation assuming that all content from a domain can be downloaded on a single TCP connection at the maximum achieved goodput seen for that domain. What is the page load time in this case?

    iii. Why is there a difference in the above two estimates?

4. Implementation

    a. Build your own web page download program which takes the HAR file or object tree as an input, and downloads the objects according to the dependency order in the object tree. Your program should download and save on disk all the objects downloaded for the web page. You can create folders for each domain name and save the objects in the appropriate folder. Clear the DNS cache before running your program. Arguments your program should accept:

        i. HAR file generated by Firebug, or object tree generated by your scripts above

        ii. Maximum number of TCP connections per domain

        iii. Maximum number of objects per TCP connection

    b. Use the Sockets library for this, and not higher layer libraries such as HTTPUrlConnection in Java

    c. Run your program with the same parameter values you discovered in part 3.c.iv above for browsers. Are your page load times similar to that of the browser?

    d. Run your program with a range of other parameter values and find the best choice which gives the minimum page load time

e. What information do you think if you had in advance, you could decide the optimal choice of parameters to use? How could the HTTP protocol, and/or the HTML standard, and/or link/network layer information available, be modified to provide this information to the browser and help it make better scheduling decisions?

Sections	Standard marks out of	Bonus marks out of
3a	10	5
3b	5	0
3c	21	11
3d	6	2
4	12	10

There are LOTS of bonus marks to be gained, please talk to your TA mentors to understand what is expected!

5. What to submit

   a. A .tar.gz file with the following folder

      i. /src containing a script (in python or perl or any other language you are comfortable with) and README file, that takes the HAR and PCAP filenames as input and produces as output the various information asked in part 3a, object and download tree CSVs in part 3b, and information asked in part 3c. You will also be evaluated on the neatness of the output produced, so spend time thinking this out.

      ii. /doc containing a pdf report on the data, analysis, and insights asked in parts 3a-3d. You should use graphs, timelines, and other tools to present your results – presentation has a high premium to clearly convey your insights

      iii. /src containing a program (in python or perl or any other language) and README file, which takes the parameters listed in part 4a and produces at least the page load time output.

      iv. /doc containing a pdf report on your results as asked in part 4, with neat graphs and/or tables that show which combination of parameters works well. Presentation of research results carries a high premium – do not claim anything that does not follow from the data, and whatever you claim should be obvious from the way you have presented the data.