Monday, February 29, 2016

Caffeine

Caffeine reduces network latency in native iOS applications.

This document empirically discusses the following two specific items:

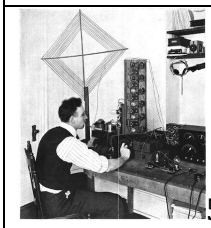
- The amount of latency reduced
- What characteristics affect latency reduction

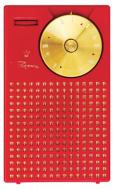
Abstract

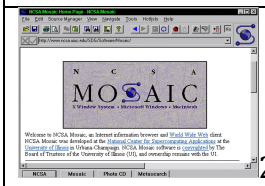
We collocate a server in Switzerland and repeatedly download images from it using an iOS Simulator. In conducting downloads of a series of images, we find that fully encrypted Caffeine has 4.5x-8.5x greater throughput than HTTPS showing a just as great decrease in perceived network latency. We find Caffeine's improvements over HTTPS to be consistent despite network congestion or otherwise adverse conditions (packet loss, lot of devices on the network, etc).

A Historical Experiment: Consumer Technology over Time

An OS X server in Zurich Switzerland contains twenty-four images of great achievement—from Edison's 1877 cylinder phonograph to Musk's SpaceX. These images total 1.615506MB. Smaller versions of these images are below.

<table>
<tr>
<td>  1</td>
<td>  2</td>
<td>  3</td>
<td>  4</td>
</tr>
<tr>
<td>  5</td>
<td>  6</td>
<td>  7</td>
<td> 

8</td>
</tr>
<tr>
<td>  9</td>
<td>  10</td>
<td>  11</td>
<td>  12</td>
</tr>
<tr>
<td>  13</td>
<td>14  </td>
<td>  15</td>
<td>  16</td>
</tr>
<tr>
<td> 

17</td>
<td>  18</td>
<td>19 
</td>
<td>  20</td>
</tr>
<tr>
<td> 

21</td>
<td>  22</td>
<td>  23</td>
<td>24 
</td>
</tr>
</table>

These twenty-four images are downloaded from the server in Switzerland to an iOS simulator in Cambridge, Massachusetts consecutively from oldest to newest (1 to 24). The sizes in bytes of each individual image are below.

1	106941
2	94482
3	5257
4	95655
5	51468
6	5240
7	166199
8	15091
9	154009
10	38772
11	12932
12	65515
13	65486
14	78480
15	14703
16	73451
17	55605
18	55750
19	16722
20	13358
21	309497
22	42669
23	71259
24	6965

To test HTTP/S downloads, we use the traditional NSURLSession provided by Apple's Foundation Library.
To test our alternative approach to mobile networking, we use Caffeine's provided APIs.

Methodology

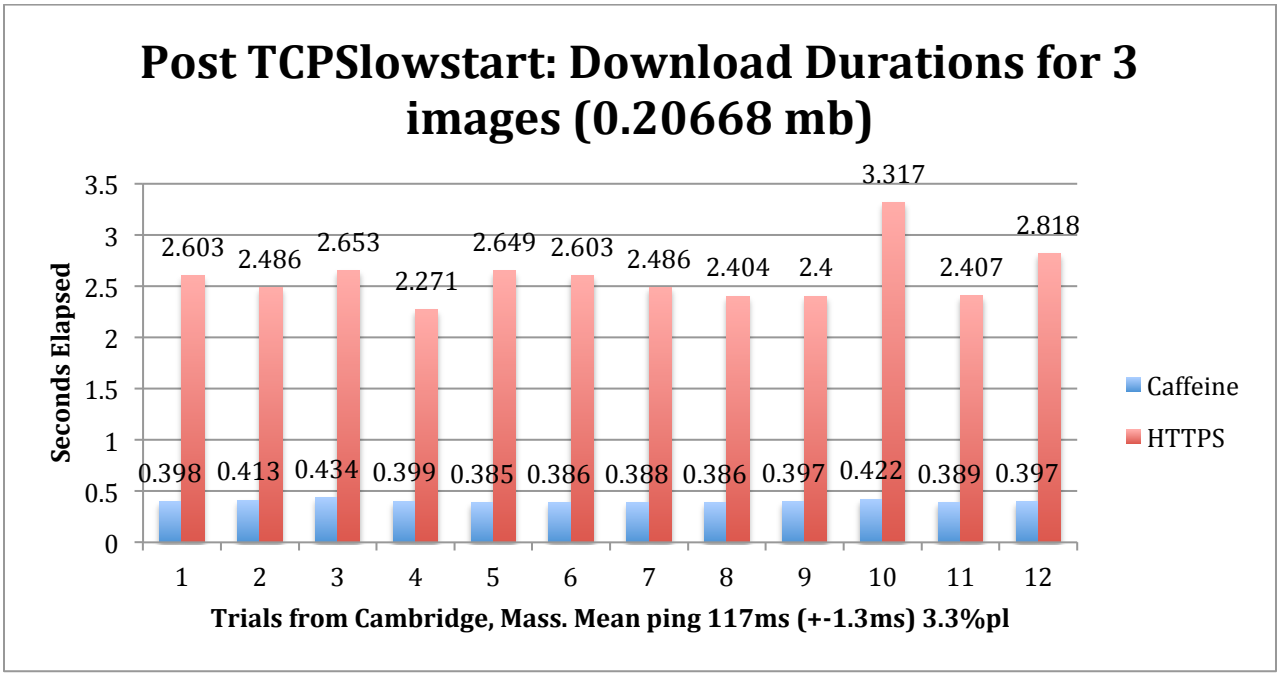
In attempt to scientifically evaluate the difference between Caffeine and HTTPS as it pertains to images, we took the following steps.

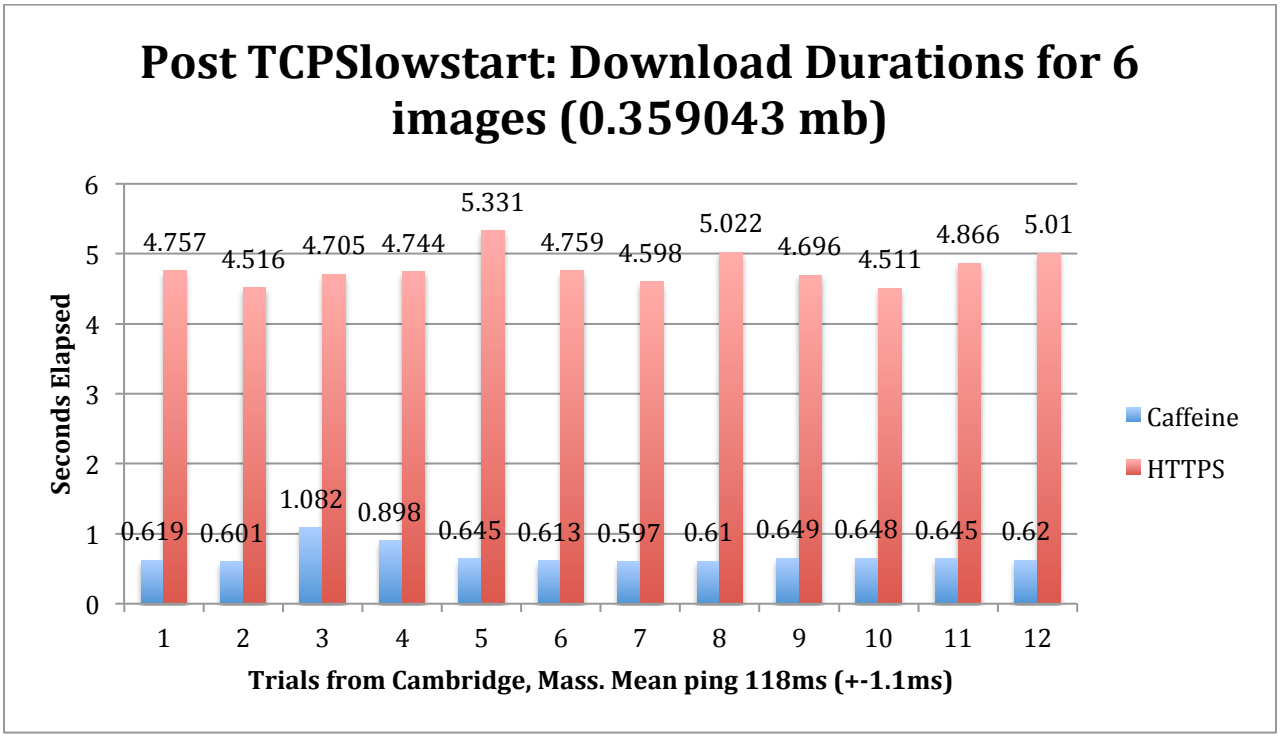
Note: All data was collected from within the same 100-meter radius including no other smartphones/laptops. All tests were run on the Harvard University secure WiFi. The results in this study may be more dramatic than if the client and server were in the same country (U.S. to U.S. connection, e.g.). We plan to test this variable in the immediate future.

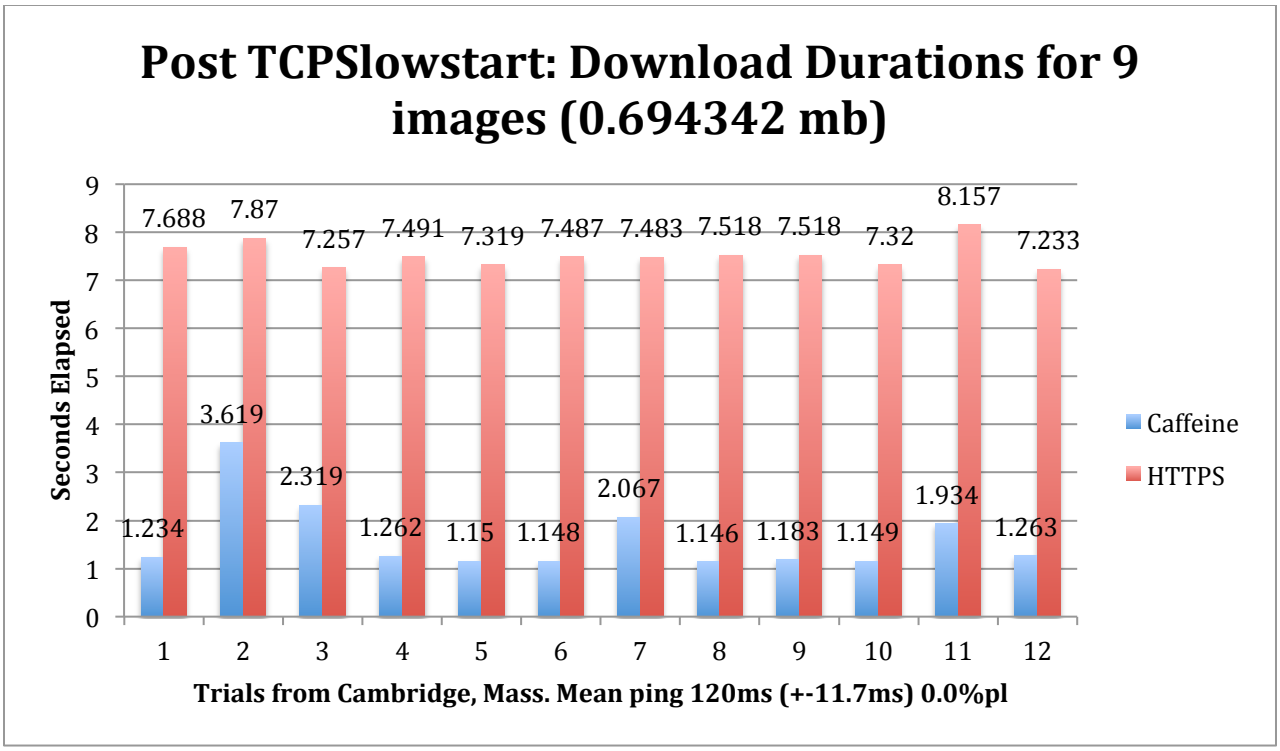
1. In Cambridge, MA, start the client on an iPhone 6 simulator running iOS 9.2 from Xcode v7.2
2. In Zurich, Switzlerland start both backend servers (Caffeine and Apache) on OS X El Capitan 10.11.3 with 2.48 GHz processing power and 2GB DRAM.
3. Ping server in Switzerland (using ICMP) 100 times and record the following:
    a. Average ping,
    b. Standard deviation of ping
    c. Packet loss.
4. Make an initial request for images from the client using Caffeine to remove TCPSlowstart.
5. Initiate Caffeine client requesting 24 images from Caffeine server 12 times with no greater than 1 second in between each of the 12 times.
6. Make an initial request for images from client using HTTPS to remove TCPSlowstart
7. Initiate NSURLSession requesting 24 images from Apache server 12 times with no greater than 1 second in between each of the 12 times.
8. Repeat steps 3-7 for a total of 10 times where the number of images requested is some different multiple of three (3, 6, 9, … 27, 30 – after 24 images, we reuse images from the beginning)

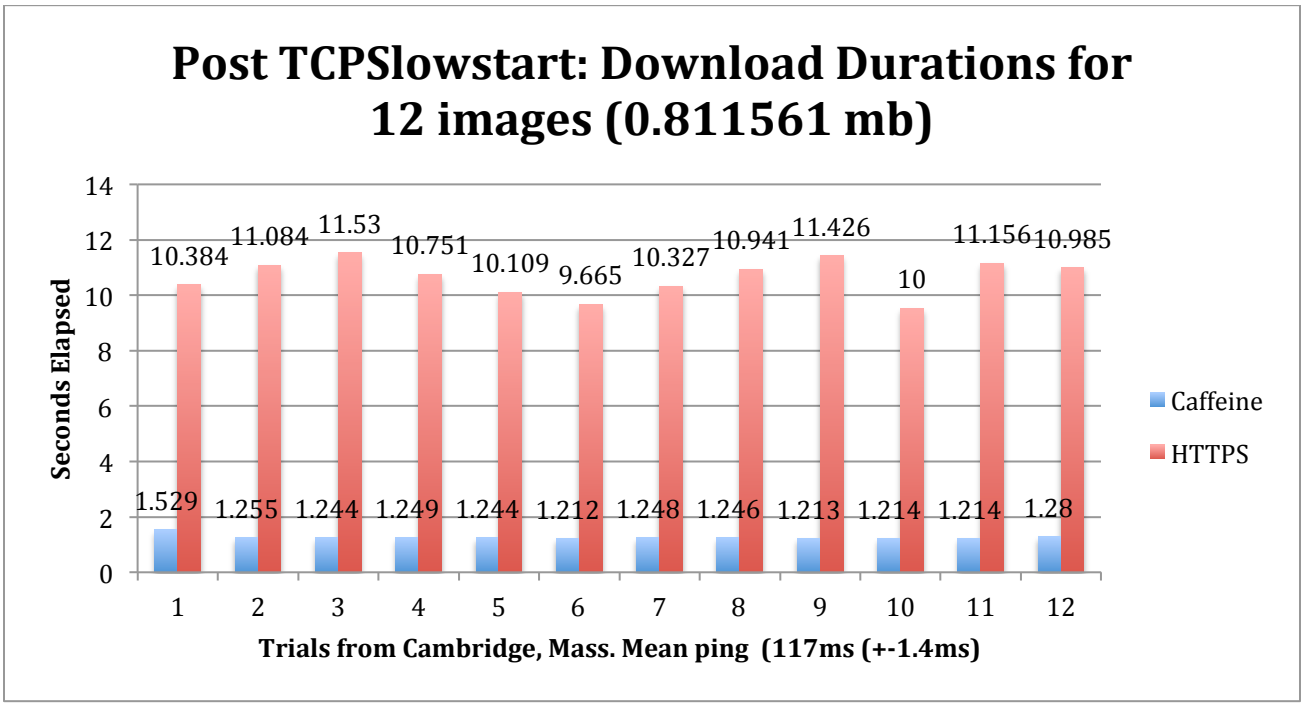
Results

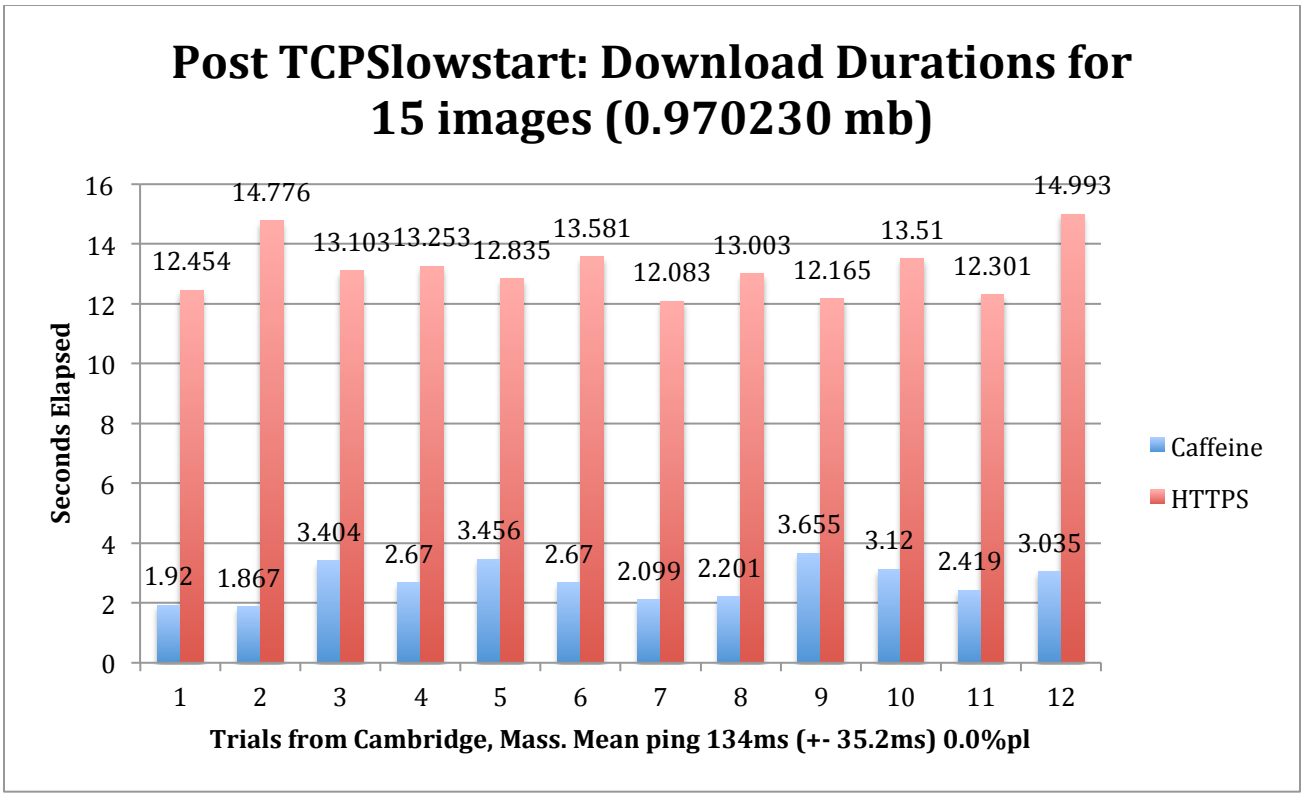
For every 10 sets (each with an additional 3 images) of trials (12 for Caffeine, 12 for HTTPS), we have the following results.

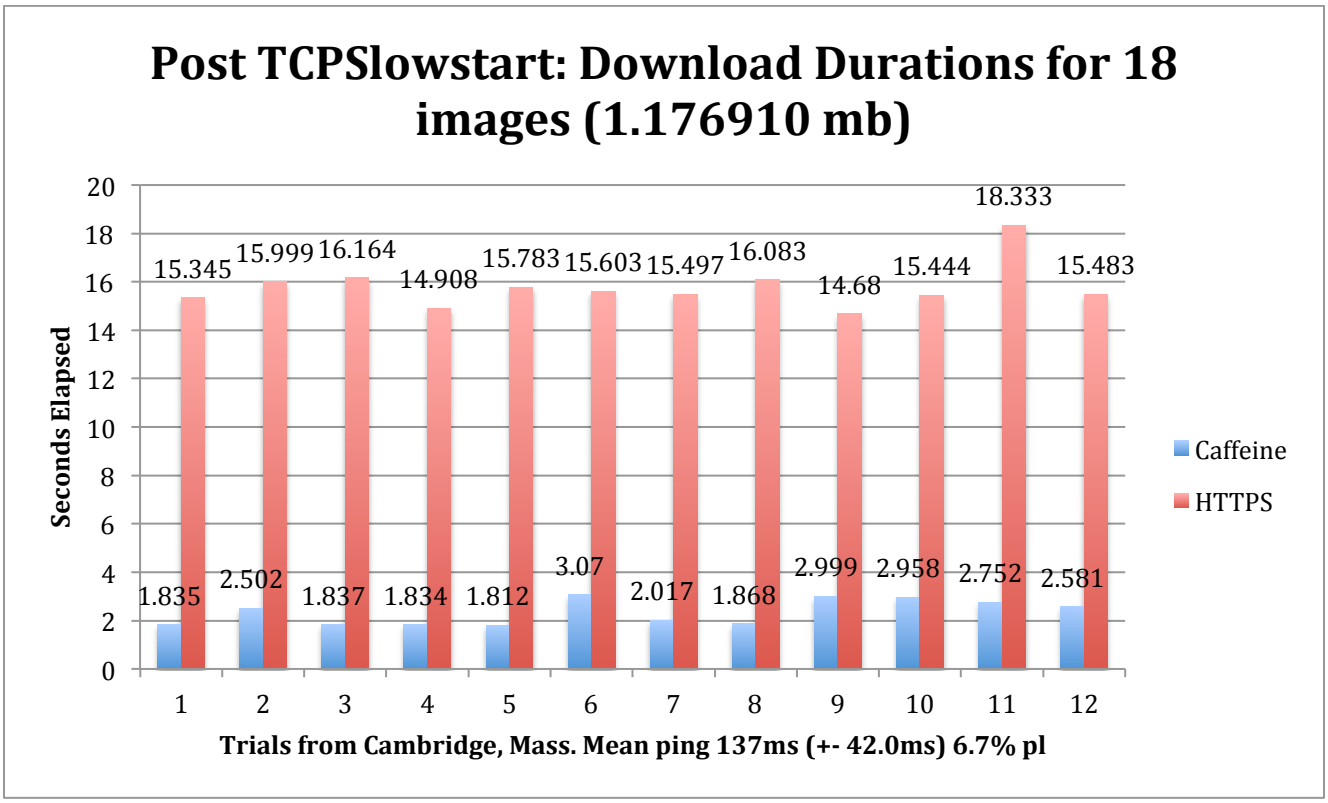


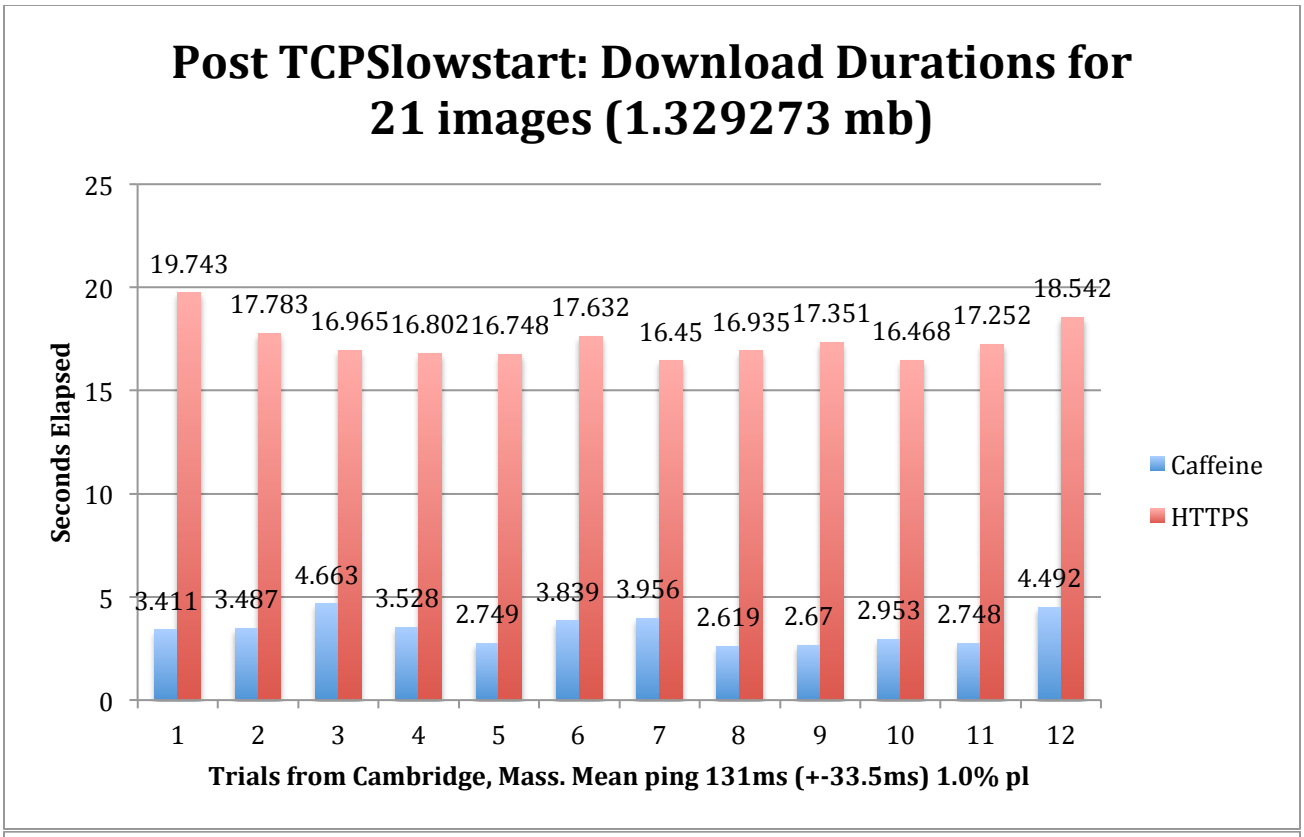


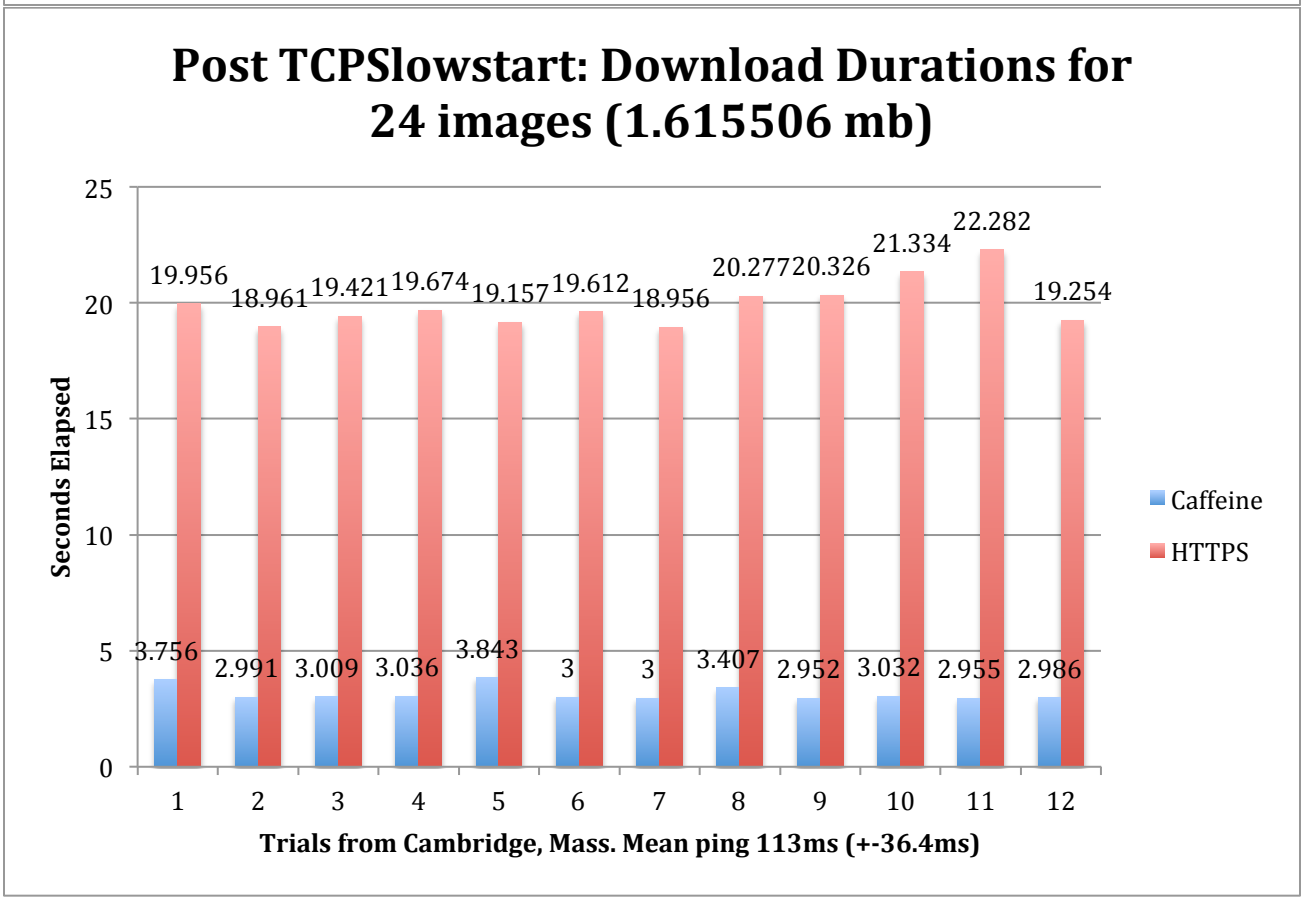

Post TCPSlowstart: Download Durations for 9 images (0.694342 mb)
Seconds Elapsed
0
1
2
3
4
5
6
7
8
9
7.688
7.87
7.257
7.491
7.319
7.487
7.483
7.518
7.518
7.32
8.157
7.233
1.234
3.619
2.319
1.262
1.15
1.148
2.067
1.146
1.183
1.149
1.934
1.263
1
2
3
4
5
6
7
8
9
10
11
12
Trials from Cambridge, Mass. Mean ping 120ms (+-11.7ms) 0.0%pl
Caffeine
HTTPS

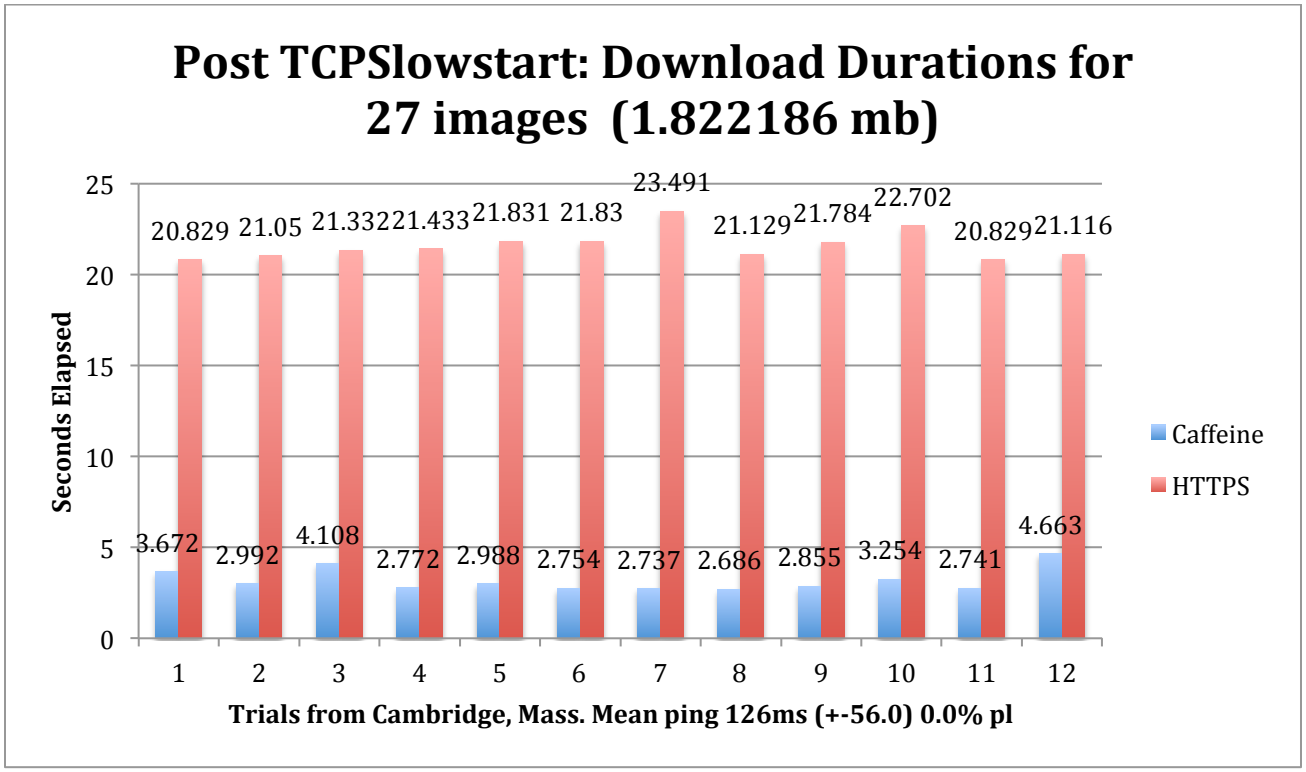

Post TCPSlowstart: Download Durations for 12 images (0.811561 mb)
Seconds Elapsed
0
2
4
6
8
10
12
14
10.384
11.084
11.53
10.751
10.109
9.665
10.327
10.941
11.426
10
11.156
10.985
1.529
1.255
1.244
1.249
1.244
1.212
1.248
1.246
1.213
1.214
1.214
1.28
1
2
3
4
5
6
7
8
9
10
11
12
Trials from Cambridge, Mass. Mean ping (117ms (+-1.4ms)
Caffeine
HTTPS

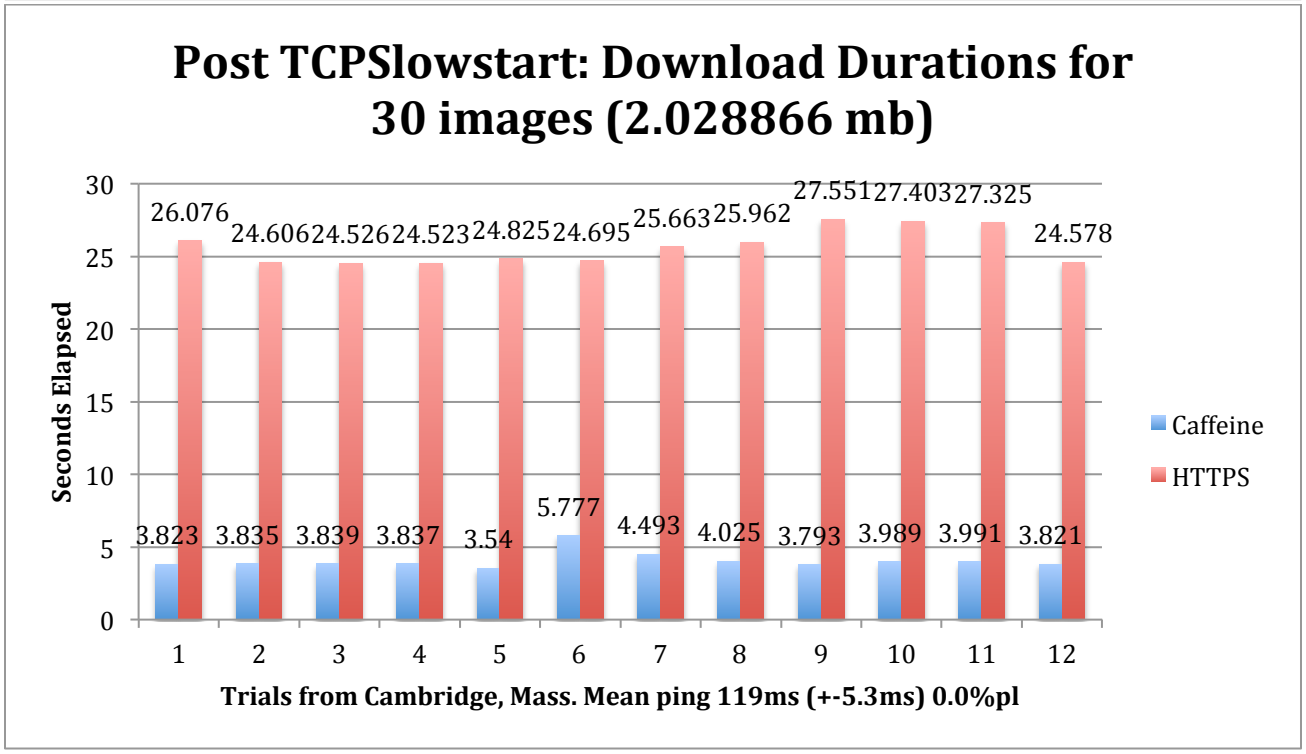

Post TCPSlowstart: Download Durations for 15 images (0.970230 mb)
Seconds Elapsed
0
2
4
6
8
10
12
14
16
12.454
14.776
13.103
13.253
12.835
13.581
12.083
13.003
12.165
13.51
12.301
14.993
1.92
1.867
3.404
2.67
3.456
2.67
2.099
2.201
3.655
3.12
2.419
3.035
1
2
3
4
5
6
7
8
9
10
11
12
Trials from Cambridge, Mass. Mean ping 134ms (+- 35.2ms) 0.0%pl
Caffeine
HTTPS


Post TCPSlowstart: Download Durations for 18 images (1.176910 mb)
Seconds Elapsed
0
2
4
6
8
10
12
14
16
18
20
15.345
15.999
16.164
14.908
15.783
15.603
15.497
16.083
14.68
15.444
18.333
15.483
1.835
2.502
1.837
1.834
1.812
3.07
2.017
1.868
2.999
2.958
2.752
2.581
1
2
3
4
5
6
7
8
9
10
11
12
Trials from Cambridge, Mass. Mean ping 137ms (+- 42.0ms) 6.7% pl
Caffeine
HTTPS

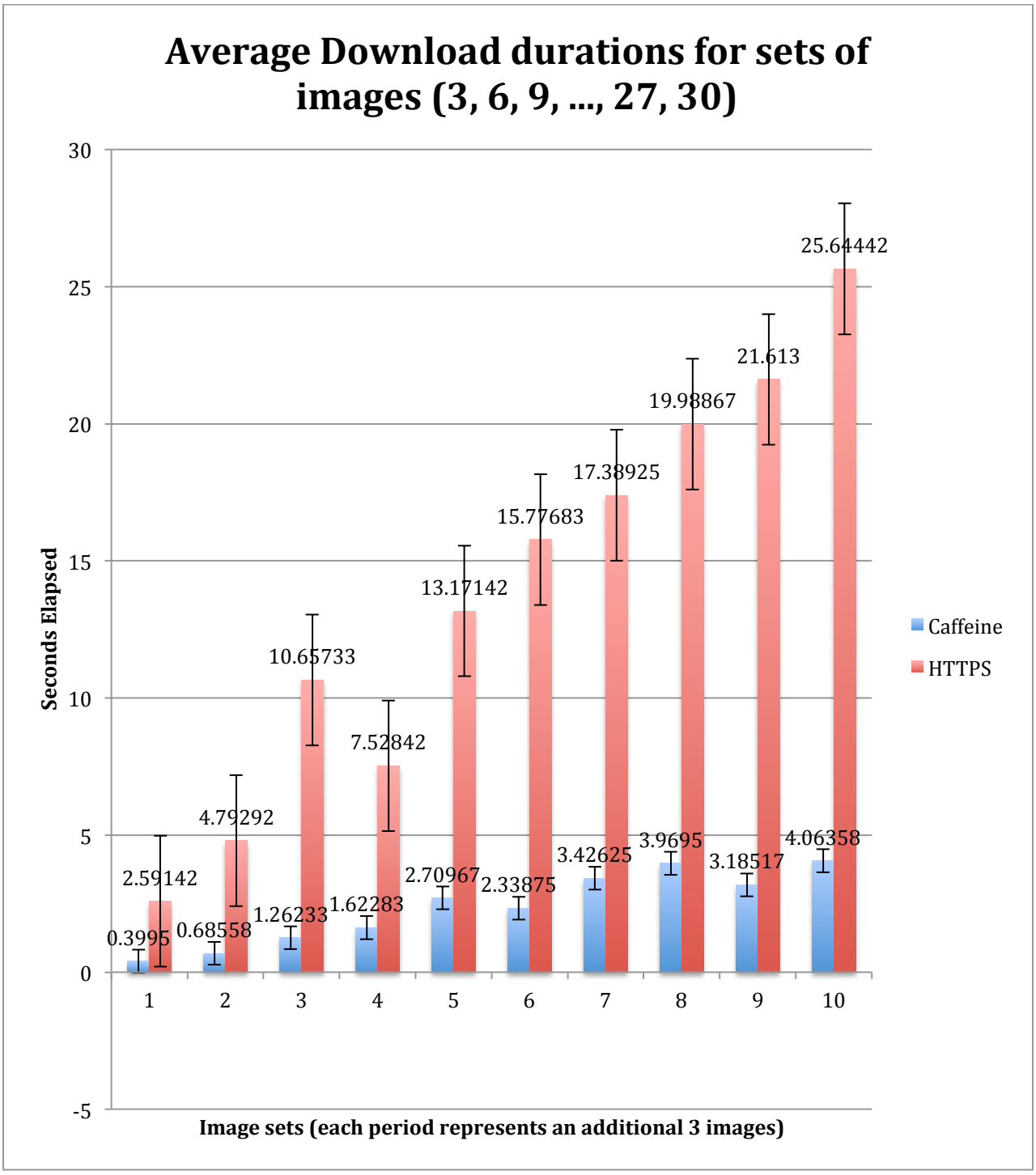

Post TCPSlowstart: Download Durations for 21 images (1.329273 mb)
Seconds Elapsed
25
20
15
10
5
0
19.743
17.783
16.965
16.802
16.748
17.632
16.45
16.935
17.351
16.468
17.252
18.542
3.411
3.487
4.663
3.528
2.749
3.839
3.956
2.619
2.67
2.953
2.748
4.492
1
2
3
4
5
6
7
8
9
10
11
12
Trials from Cambridge, Mass. Mean ping 131ms (+-33.5ms) 1.0% pl
Caffeine
HTTPS


Post TCPSlowstart: Download Durations for 24 images (1.615506 mb)
Seconds Elapsed
25
20
15
10
5
0
19.956
18.961
19.421
19.674
19.157
19.612
18.956
20.277
20.326
21.334
22.282
19.254
3.756
2.991
3.009
3.036
3.843
3
3
3.407
2.952
3.032
2.955
2.986
1
2
3
4
5
6
7
8
9
10
11
12
Trials from Cambridge, Mass. Mean ping 113ms (+-36.4ms)
Caffeine
HTTPS

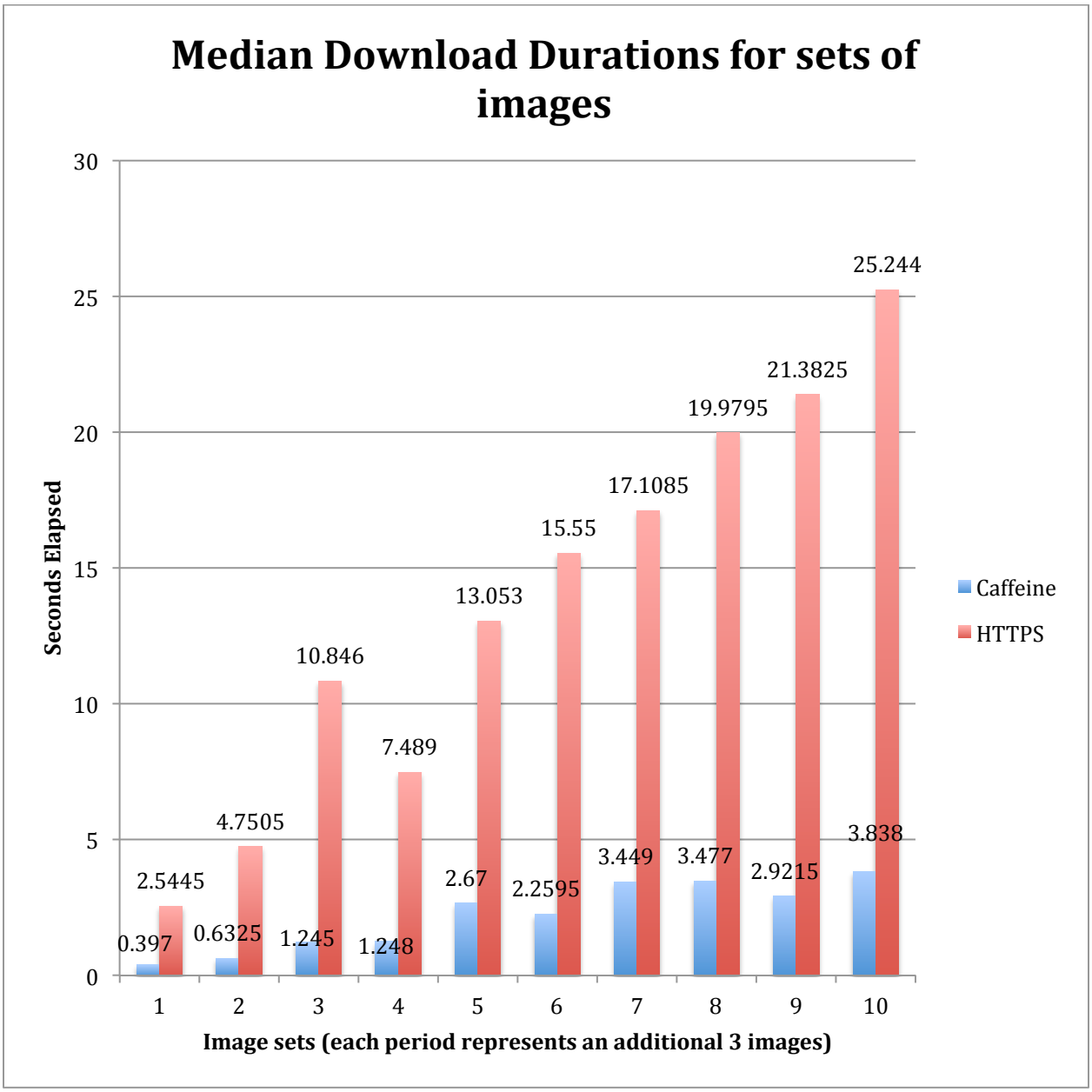

Post TCPSlowstart: Download Durations for 27 images (1.822186 mb)
Seconds Elapsed
0
5
10
15
20
25
20.829 21.05 21.332 21.433 21.831 21.83 23.491 21.129 21.784 22.702 20.829 21.116
3.672 2.992 4.108 2.772 2.988 2.754 2.737 2.686 2.855 3.254 2.741 4.663
1 2 3 4 5 6 7 8 9 10 11 12
Trials from Cambridge, Mass. Mean ping 126ms (+-56.0) 0.0% pl
Caffeine
HTTPS


Post TCPSlowstart: Download Durations for 30 images (2.028866 mb)
Seconds Elapsed
0
5
10
15
20
25
30
26.076 24.606 24.526 24.523 24.825 24.695 25.663 25.962 27.551 27.403 27.325 24.578
3.823 3.835 3.839 3.837 3.54 5.777 4.493 4.025 3.793 3.989 3.991 3.821
1 2 3 4 5 6 7 8 9 10 11 12
Trials from Cambridge, Mass. Mean ping 119ms (+-5.3ms) 0.0%pl
Caffeine
HTTPS

A cross-sectional look at the average results:



A cross-sectional look at the median results:

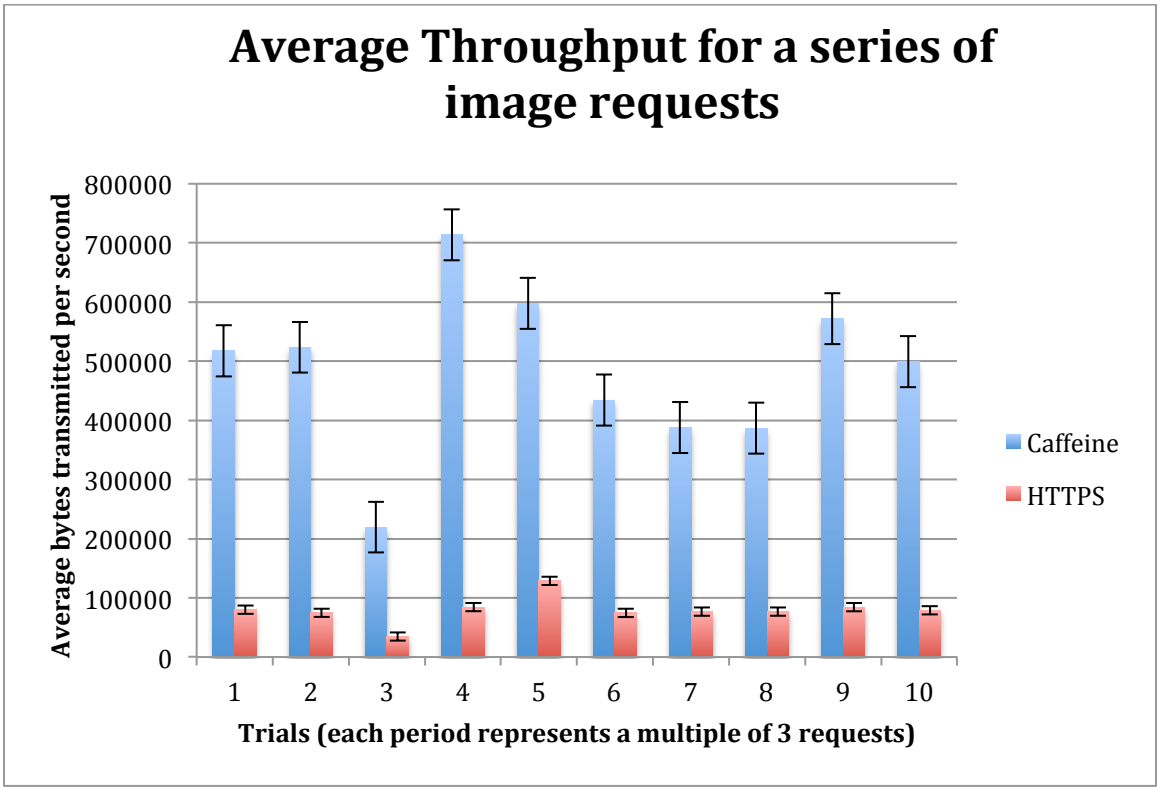


On average, Caffeine is anywhere from 400% - 850% faster than HTTPS. There are some anomalies in the data such as the 4th period in the median graph (above). This can be explained by waves of congestion on the network, or latency overseas. Across the board, the improvement gains of Caffeine over HTTPS remain relatively consistent.

Because each period receives an additional 3 images, their total data size does not increase linearly like the total number of requests. For example, an additional 3

images fetched from the server always results in 3 additional requests, but the amount of total data for each of those requests is variable.

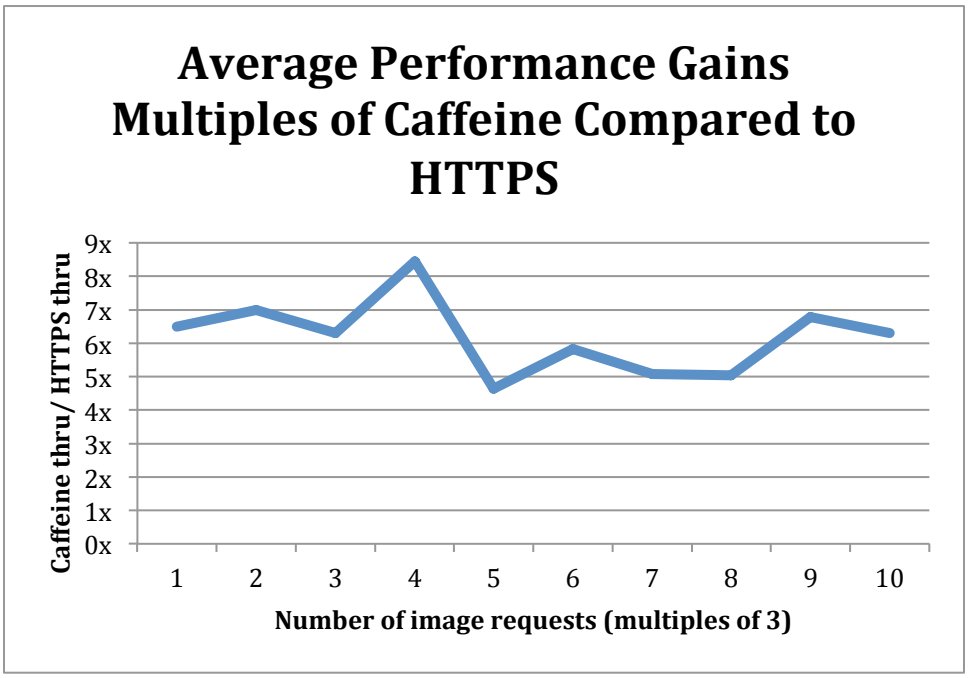
To normalize this variation, we determine the average throughput (bytes/second).



This graph much more clearly displays the difference of Caffeine. It describes how much data can "fit through the pipe" per second. On the X-axis, the total number of requests increases by 3 periodically.

In times of network stress or congestion, such as in the 3rd period above (9 requests), the performance of HTTPS prorates with the performance of Caffeine. This suggests that Caffeine's performance relative to HTTPS remains consistent despite unpredictable networks phenomena (severe weather, congestion, miscellaneous latency).

The percentage gain of Caffeine versus HTTPS is shown by the graph below.



The minimum average multiplier improvement of Caffeine over HTTPS shown by this graph is ~450%, while the maximum is ~850%.

Closing Remarks

When developing Caffeine, we continually asked what is actually necessary for iOS application networking environments. This approach has resulted in the reduction of a lot of (what turn out to be) unnecessary features/data in HTTP. The amount of bloat and latency incipient to HTTP surprised us quite a lot.

Forthcoming research include quantitative analyses of resource usage—determining how much bandwidth, CPU, battery-life would be saved in a Caffeinated environment.

Caffeine, as far as we know (and we've looked), is the fastest way to send data in iOS request-response architectures. We know this is the future of networking, and we are looking for those intrepid early adopters.

Email us
beyondjava@caffei.net

The future is already here — it's just not evenly distributed.
—William Gibson