How to determine latency of a remote server through the browser

Asked 15 years, 10 months ago Modified 4 years, 4 months ago

Viewed 19k times

18

I run a couple of game tunnelling servers and would like to have a page where the client can run a ping on all the servers and find out which is the most responsive. As far as I can see there seems to be no proper way to do this in JavaScript, but I was thinking, does anybody know of a way to do this in flash or some other client browser technology maybe?

javascript flash browser latency ping

Share

Improve this question

Follow

edited Feb 6, 2009 at 7:41

asked Feb 2, 2009 at 13:25

Mladen Mihajlovic

6,415 ● 8 ● 42 ● 58

I'm quite interested about your solution. Any chance to get some results ? Thanks in advance ! – Theo.T Mar 22, 2009 at 23:26

The most interesting answer to me was the one by Eliram (stackoverflow.com/questions/503199/…), but StackOverflow already chose the answer for me. I haven't persued this project any further though. I'm not a flash dev
– Mladen Mihajlovic Mar 23, 2009 at 10:10

9 Answers

Sorted by: Highest score (default)

18

+50

Most applet technology, including Javascript, enforces a same-origin policy. It may be possible to dynamically add DOM elements, such as images, and collect timing information using the onload event handler.

Psuedo-code

```
for (server in servers) {
  var img = document.createElement('IMG');
  server.startTime = getCurrentTimeInMS();
  img.onload=function() { server.endTime = getcurrentT
  img.src = server.imgUrl;
}
```

Then wait an appropriate time and check the timing for each server object. Repeat as needed and compute averages if you want. I'm not sure what kind of accuracy you can expect.

Disadvantages:

- You are probably using the wrong tool for the job. A browser is not equipped for this sort of application.
- It's probably quite inaccurate.

- If the resource you request is cached it won't give you the results you want, but you can work around that by changing the url each time.
- This is bandwidth-intensive compared to a normal ping. Make the image tiny, such as a spacer.gif file.
- The timing depends not only on the latency of the remote server but the bandwidth of that server. This may be a more or less useful measure but it's important to note that it is not simply the latency.
- You need to be able to serve HTTP requests from the various servers and, crucially, each server should serve the exact same resource (or a resource of the same length). Conditions on the server can affect the response time, such as if one server is compressing the data and another isn't.

Share Improve this answer Follow

answered Feb 18, 2009 at 16:44

Mr. Shiny and New 安宇
13.9k ● 6 ● 46 ● 65

Use new Image() instead of creating a whole DOM-object.
– Georg Schölly Feb 21, 2009 at 21:44

@gs: Is there an advantage to using new Image()?
– Mr. Shiny and New 安宇 Feb 23, 2009 at 14:43

The problem with all these solutions which include downloading something is that it's really testing the speed of the web server, and not the latency of the network connection. So it's not really a true reflection, especially for my needs which are for very specific network latency speeds.
– Mladen Mihajlovic Mar 23, 2009 at 10:11

@Mladen Mihajlovic: This is the limitation of the technology you are using. Most applet technology doesn't allow you to make arbitrary TCP/UDP connections to arbitrary servers. That's what you'd need. – Mr. Shiny and New 安宇 Mar 23, 2009 at 12:20

It seems that it's possible in Flash (see answer below). Only problem is I don't know flash at all :) – Mladen Mihajlovic Mar 25, 2009 at 17:19

9

Before the call to the server, record the Javascript time:

```
var startTime = new Date();
```

Load an image from the server:

```
var img = new Image()
img.onload = function() {
    // record end time
}
img.src = "http://server1.domain.com/ping.jpg";
```

As soon as the request is finished, record the time again. (Given of course that the request didn't time out.)

```
var endTime = new Date();
```

Your ping in milliseconds is:

```
var ping = endTime. getTime() - startTime.getTime();
```

Share Improve this answer Follow

edited Feb 21, 2009 at 21:24

answered Feb 21, 2009 at 21:12

Georg Schölly

126k ● 54 ● 223 ● 276

6

All you really need is the time from the connection start, to the time of the first readystate change...

```
function getPing() {
  var start;
  var client = getClient(); // xmlhttprequest
object
  client.onreadystatechange = function() {
    if (client.readyState > 0) {
      pingDone(start); //handle ping
      client.onreadystatechange = null; //remove
handler
    }
  }

  start = new Date();
  client.open("HEAD", "/ping.txt"); //static file
  client.send();
}

function pingDone(start) {
  done = new Date();
  ms = done.valueOf() - start.valueOf();
  alert(ms + "ms ping time");
}

function getClient() {
  if (window.XMLHttpRequest)
    return new XMLHttpRequest();

  if (window.ActiveXObject)
```

```
    return new
ActiveXObject('MSXML2.XMLHTTP.3.0');

  throw("No XMLHttpRequest Object Available.");
}
```

Share Improve this answer Follow

edited Feb 21, 2009 at 21:42

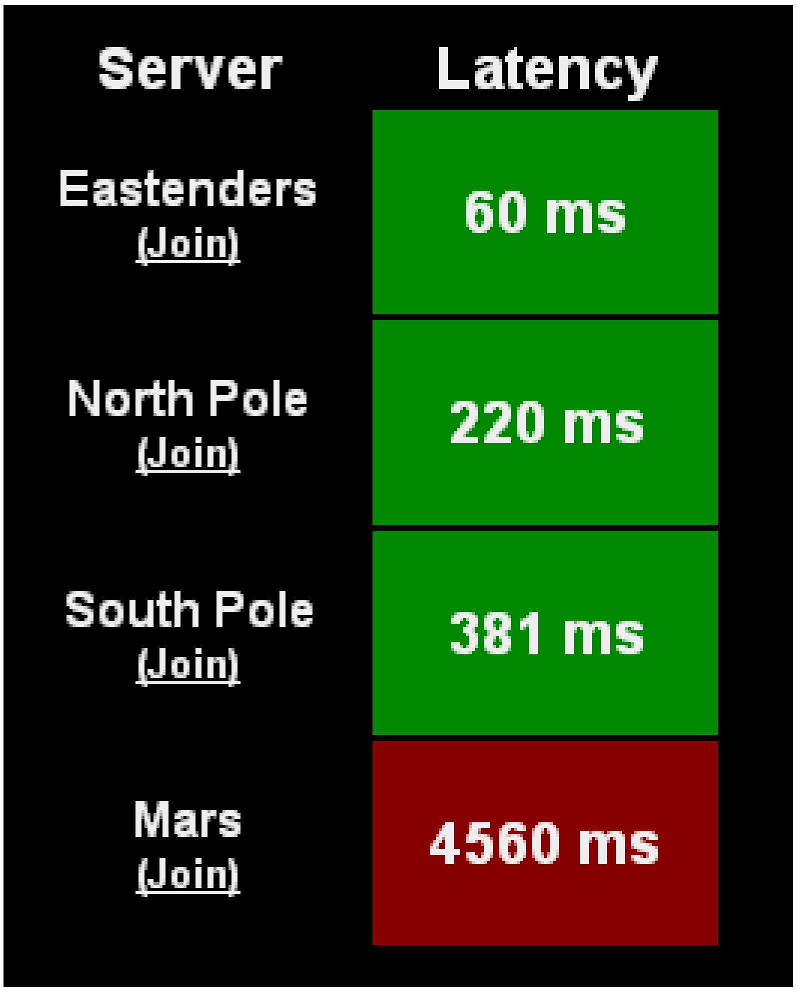
answered Feb 21, 2009 at 21:31

Tracker1
19.3k ● 12 ● 85 ● 106

This will only work if you are pinging the server hosting the webpage, and not for pinging other servers.
– Mr. Shiny and New 安宇 Feb 24, 2009 at 15:18

4

Here's an `<iframe>` approach:



Server	Latency
Eastenders (Join)	60 ms
North Pole (Join)	220 ms
South Pole (Join)	381 ms
Mars (Join)	4560 ms

(source: magnetiq.com)

Create a table (not necessarily in the literal `<table>` sense) with two columns. The first column will hold the name of servers (and possibly links to them). The second column has iframes that load probe documents from the respective servers. Each probe document does this on the initial fetch request:

1. Get current system time
2. Do a redirect (302) to a second probe document while passing the system time as a query parameter
3. The second probe document reads the current system time, calculates the delta from the initial reading that was passed to it and just displays it in big fat letters. This delta will be the time it took for the

> server to respond to the client with a redirect response plus the time it took for the client to make the second request to the redirection target. It's not exactly a "ping" but it's a comparable measure of the client's relative latency with each server. In fact, it's a "reverse ping" from the server to the client.

You'd be using iframes without infringing the same-domain policy because there's no attempt at manipulating the iframe contents at all. The player will simply see the values with his/her own eyes and you'll rely on the user glancing at the numbers and clicking on the server link that makes the most sense.

Share Improve this answer Follow

edited Mar 3, 2019 at 14:00
Glorfindel
22.6k ● 13 ● 89 ● 116

answered Feb 24, 2009 at 20:28
Ates Goral
140k ● 27 ● 141 ● 191

3

Anything that makes an HTTP request (like most of the answers here) will generally measure a latency that's at least twice of what you'd see for a normal ping, because you'll need the three way handshake and the termination packet at minimum (two round trips rather than one). If you make HTTP requests, try to keep the headers to a minimum. A long enough header (due to a chatty server, or cookies etc on the client) can add additional round trips into the mix, throwing off your measurements.

As Cherona points out, if you already have an active HTTP 2 connection to the server, or if the server speaks HTTP 3, then this may not be the case.

The most accurate option would be to open a websocket connection to each server and measure the time it takes to send a tiny message and receive a tiny response (after the connection has been established).

Share Improve this answer Follow

edited Jul 31, 2020 at 2:57

answered Feb 25, 2009 at 4:28

Peter Burns
45.3k ● 7 ● 40 ● 56

1 This isn't true for HTTP2 and HTTP3 – Cherona Jun 30, 2020 at 15:38

Good point! Edited to add mention of HTTP 2, HTTP 3, and websockets. – Peter Burns Jul 31, 2020 at 2:57

2

If you are talking about running something client side, I am not sure this is possible due to security reasons.

Maybe your best bet would be a java applet - but again this needs to be checked against local security policy.

If I try to think about some hack in JS to do this, maybe you can try to send an async request with a callback function which measures the milliseconds it took - but this is just off the top of my head.

Share Improve this answer Follow

edited Feb 2, 2009 at 13:44

answered Feb 2, 2009 at 13:32

Yuval Adam
165k ● 95 ● 315 ● 404

I know one way I can do it is by sending a HEAD request, but I was more hoping that there is a way to do it using flash? – Mladen Mihajlovic Feb 6, 2009 at 7:42

Java applets are only allowed to contact their originating server. – Magnar Feb 18, 2009 at 9:58

Applets with more permissive security are allowed if they are signed and accepted. Or at least they used to be. The last time I saw one was with Netscape 4. – Zan Lynx Feb 25, 2009 at 2:10

2

It's not that hard to measure server response time in Flash.

Flash must ask for a policy file before accessing remote servers. The default location for such policy file is at the root folder of the server: /crossdomain.xml

(You can easily find information about the crossdomain file format)

Since such file is needed anyway, why not use it to measure server response time? Load the file itself instead of an image and measure the time it took using getTimer() .

This will give you a good estimate on HTTP connections.

But if you're dealing with game servers, you might want to directly check the speed of the TCP connection. To do that you'll need to use the flash.net.Socket You'll also have to ask for a policy file first by running: Security.loadPolicyFile("xmlsocket://server.domain.com:5342");

Where 5342 represents your server's port number where it should respond with the proper XML policy string. After making the socket connection, any request/response will let you measure different server response times.

Share Improve this answer Follow

answered Feb 22, 2009 at 19:29

Eliram

616 ● 2 ● 9 ● 21

This sounds quite promising. Do you have some code, or an example flash "applet" I can see? I'm not an actionscript programmer myself. – Mladen Mihajlovic Feb 23, 2009 at 8:49

The problem with 'file pings' is that you would evaluate the http server response whereas your target resource for

1

the games you serve may have a very different behavior and thereby a different latency.

Just an idea out of the blue, maybe even unrealistic depending on the actual context: but, wouldn't it be interesting to make a server script based on a short sequence of tasks typically executed by the servers during the gameplay (e.g. opening a RTMP connection, retrieving an information, sending it back). Depending on the total number of servers, you could almost opening them simultaneously and define the first response as winner (subtracting the time your client requires independently to process each query).

Of course this is a quite expensive method server-side-speaking, but at least you would hopefully get a reliable result (server and network latencies summed up). Even if it takes a couple seconds to evaluate, this would be the matter of a fraction of the total enjoyable game-play.

Share Improve this answer Follow

answered Feb 21, 2009 at 20:59

Theo.T
9,227 ● 4 ● 25 ● 39

Based on the responses of @Mr. Shiny and @Georg Schölly , a complete and commented example.

1

In order to test, just copy and paste the codes below in the same order, in a empty .html, .php or other compatible file.

Before start the get, record the current Javascript time. Using `new Date()`, we create a new date object with the current date and time.

```
<script type="text/javascript">

var startTime = new Date();
```

Now let's create a html object image, still without source, and attribute it to the variable img.

```
var img = new Image();
```

The next spet is put a source in the image. The .src reflects the src html attribute. Important! Point your img.src to a very small and lightweight image file, if possible anything less than 10KB.

To prevent cache a random parameter was added at the end of file, after the .png extension.

```
var random_string = Math.random().toString();
img.src = "http://static.bbci.co.uk/frameworks/barlesq
blocks-dark.png" + "?" + random_string;
```

Now we may call our function which will run just when the image loads, because the .onload:

```
img.onload = function() {
    var endTime = new Date();
    var ping = endTime. getTime() - startTime.getTime(
    alert(img.src + " loaded in " + ping + " ms");
}
</script>
```

Inside the function we have the variable endTime that receives a date time after the source image was loaded.

Lastly, the ping variable receives the initial time minus the final time. The alert popup shows the result.

Share Improve this answer

Follow

answered Mar 29, 2019 at 19:42

Fellipe Sanches

8,095 ● 4 ● 33 ● 34