# Community-Driven Latency Measurements in Geographically-Distributed Networks

Topher White
ITER Organization
CS 90 046, St. Paul-lez-Durance
13067, France
+33 4 42 17 66 19
topherwhite@gmail.com

Dr. Antony Harfield
University of Warwick
Coventry
CV4 7AL, United Kingdom
+44 24 76 52 80 43
antonyharfield@gmail.com

Dr. Philip Tsang
Open University of Hong Kong
30 Good Shepherd St.
Hong Kong
drphiliptsang@gmail.com

## ABSTRACT

We sought a method of collecting, en masse, connection diagnostics between geographically-distributed client access points—as opposed to latency data between major server nodes. In response, this paper details the creation and launch of a web-based, client-driven, extendable database of client-to-client network latency data (ping data). It demonstrates the potential for distributed processing and data collection contained entirely within the web browser using asynchronous scripting techniques. The paper offers preliminary visual interpretations of an initial global latency dataset.

## Keywords

geolocation, open source, latency, semantic web, community

## 1. INTRODUCTION

In web-based communities, the increasing irrelevance of a client's geographic location is much-touted, generally undervaluing the dependence of every data network upon physical infrastructures. While connection speed/latency between client-and-server, or between peer-and-peer, is the parameter of greatest importance, connection quality is often erratic outside of major nodes. The *ITER Organization* is a rurally located laboratory, whose users are distantly--and also often rurally--located worldwide. As we are funded and supported by 35 different countries, our resources and our staff are spread distantly throughout the globe, amongst these member nations. We wished to measure the impact of this distributed mode of collaboration in order to ensure accountability and preparedness.

We found a lack of statistically significant data showing the correlation, or lack thereof, between physical location and client-to-client connection latency. We wanted an open database providing historical and contemporary connection latencies between precise geocoordinates. To maximize its utility, such a database must be easily accessible via a straightforward, stable, and well-documented API. To avoid privacy concerns, all data must be received and stored anonymously, with no record of the user's actual activity—only record of the activity *induced* by the experiment. Furthermore, we wished to de-centralize the data-generation, since measurements of connection latency between clients and major nodes (such as central web-servers) are already plentiful.


*Web Science Conf. 2010*, April 26-27, 2010, Raleigh, NC, USA.

Building on the procedural foundations developed for aerial WiFi surveys carried out by the *Open University of Hong Kong* in 2008 [1], we have constructed and launched a fully functional and robust system with the potential for extremely high data throughput. The efficacy of this initial launch is so encouraging that it provides grounds for imagining an entirely new method of data delivery, wherein web-content is distributed into the caches of reliable clients, who could then fulfill content requests as a moderated "swarm."

More conservatively, the data already amassed offers an invaluable depiction of the interconnected globe, sure to be useful in a handful of applications, from visual mashups to network analysis.

The aforementioned system (induced pinging amongst independent clients)--developed for this paper--shall be henceforth (informally) referred to as the Distributed Ping Project (DPP).

## 2. METHODS

### 2.1 Basic Components

There are three necessary components in the procedure:

1. a cloud-based **web-service**, acting primarily as a storage repository, directing the process and storing the results
2. a **client** "pinger" which initiates the process
3. a collection of **targets** (IP-addresses)

**Clients** are currently divided into three categories, representing three separate (though consistent in the aforementioned procedure) development tracks:

1. a daemon process running on UNIX-based **smartphones** (iPhone OS, Android OS)
2. a daemon process running on UNIX-based **laptops** (OSX and Linux-variants)
3. Javascript code snippet embedded into web pages by a webmaster, executed by clients' **browsers** during moments of user inactivity.

This paper will focus exclusively on the third type of client (passively executed browser code) since it was developed exclusively by the authors, and because it demonstrates the most prolific and promising results. A step-by-step explanation of this "third" method follows:

## 2.2 Basic Procedure

1. The **client** loads a web page on a remote web site (which has chosen to participate in the DPP) (Figure 1, Marker A). The loaded web page contains a <script> reference to a Javascript file located on the central DPP **web-service**. Upon page-load, the Javascript initializes an event listener, which awaits a period of user inactivity, in order to minimize any effect upon the user.

2. When (and if) this period of user inactivity occurs, the process awakens, and initiates a check-in with the DPP **web-service** (Figure 1, Marker B). If available, the **client** transmits its geo-coordinates concurrently.

3. The DPP **web-service** acknowledges the client's check-in, and saves the client's external IP-address and geo-coordinates. The **web-service** then browses a list of IP-addresses of other clients that have checked-in to the **web-service** within the previous 6 hours, and chooses one (based, as well, on several additional factors, which will be discussed in Section 2.3.3). The **web-service** returns this IP-address to the **client** (method of return to be discussed in 2.3).

4. The **client** receives the IP-address (Figure 1, Marker C) and passively executes a pre-determined number of "pings" to the target IP-address (method of "pinging" to be discussed in 2.3) (Figure 1, Marker D).

5. Following the pre-determined number of pings, the **client** calculates the average, standard deviation, and success rate of the resultant dataset and returns these values to the DPP **web-service**. (Figure 1, Marker E).

6. The **web-service** saves this data alongside the client's (ping source) and the target's (ping destination) IP-addresses before choosing/returning a new target IP-address to the client. Thus, **the process begins anew from step 3** and loops indefinitely until the browser is closed, or the user becomes active again.

## 2.3 Special Considerations

Outside of the browser environment, pinging hosts and interactiing with remote web-services is a straightforward and easily executed procedure. Inside the browser, however, there are a variety of protocol requirements and security rules that greatly complicate the task, and force alternative approaches.

### 2.3.1 Interacting with the DPP web-service

All modern browsers explicitly ban cross-domain scripting, such that (amongst other effects) AJAX calls may not be made to domain names other than those that are serving the current webpage. This poses a significant problem for the DPP, which relies entirely upon the client's ability to send data *and* receive target IP-addresses from the DPP web-service—a completely different domain from the web-site initially visited by the user. Browsers do, however, allow remotely located images. Thus, by disguising the web-service as a GIF image (http://www.distributedpings.org/x.gif), the browser is capable of loading it on command. Passing data into the web-service is thus quite straightforward, since they may be appended to the URL as GET variables (similar to the method employed by Google Analytics [2]). The data passed back to the browser needs to be

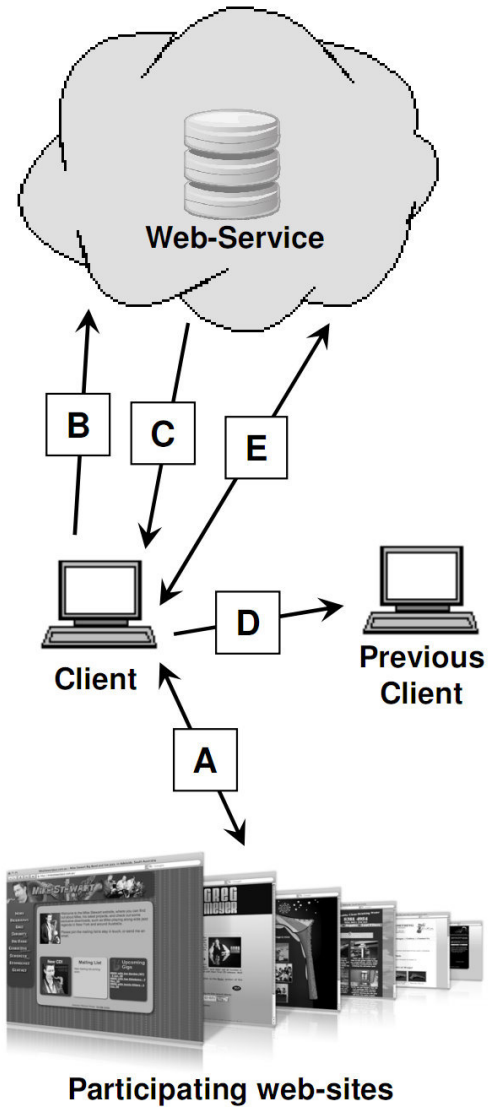


**Figure 1. Summary of client - web-service interactions.**

interpreted as an IP address, however, it must be somehow embedded into a GIF image in a manner that is parseable by the Javascript running in the client's browser. This is accomplished by calling the web-service twice (state is conserved across web-requests by using a 24-character "id"), and parsing the width and height (in pixels) of the two resultant images. The 4 integers are then assembled to reveal the target IP-address (Figure 2).

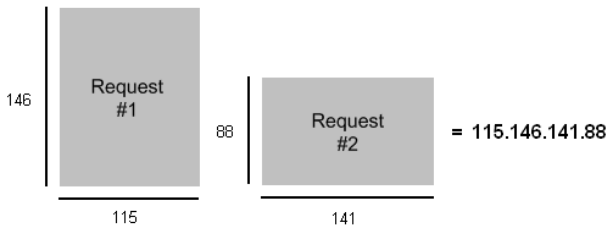


**Figure 2. Determining target IP-address from width and height of two images, provided by DPP web-service.**

### 2.3.2 Simulating "pings" within the browser

At the time of writing, no modern browser includes a mechanism for "pinging" by Javascript or Flash. Several ActiveX solutions have been built and released, but we found such an option to be unacceptable since it would require a break from the open-standard model, would limit our usable browser base to Internet Explorer users, and worst of all, would require active acceptance (to install the control) by the client.

The act of "pinging" is, in effect, a non-specific TCP connection to a given address. Even if the computer residing at a given IP address does not contain any server processes or open ports, a ping should still be returned (excluding computers in stealth mode). Given this, we were able to simulate "pinging" using the following procedure:

As with the web-service interaction, we treat each "ping" as the loading of an image in Javascript. Given a target IP (64.235.52.190, for example), we tell the browser to load an image that we presume does not exist, at an IP-address that we presume is not a web server (http://64.235.52.190/random_string.gif). If this IP is reachable, and is not in stealth mode, then requesting this non-existent URL will return an error, and the time lapse between the request and the error is equivalent to the latency time of a ping. If the IP-address does not exist, is unreachable, or is in stealth mode, then the request will time out. Thus by requesting a given image repeatedly, we may effectively simulate the act of "pinging" in the browser.

### 2.3.3 Quality Control and Garbage Collection

In our model, data is generated by a moderated "swarm", in which a large number of widely distributed worker processes bear the load, while a single centralized location collects and directs the masses. Making this direction more intelligent and efficient is extremely important in maintaining and improving the integrity of the experiment and its dataset. Each client should be identical and mindless.

Since target IP-addresses are the only information returned to the clients, we want to make sure that these IPs will yield results. It is therefore very important to rate the quality of a given IP before returning it. This begins with the recording of failed pings. Each time a client fails to return a valuable data set (2 pings or less), the target IP is penalized by one "point." Any IP with more than 8 "points" becomes disqualified, and is not returned by the web-service.

Expiration is another concern. Most home-bound broadband connections use dynamic IP addresses, which drift somewhat erratically. The web-service maintains the freshness of target IPs by only returning addresses that have, themselves, been clients within the previous 6 hours.

Even with these two quality control measures, some IPs are unfairly penalized by failed connections from the source. If a given IP has been continuously active for many hours, and has successfully returned hundreds of pings from a diverse set of sources, but has also had 8 failures, we do not wish for it to be disqualified. Thus, the web-service will also return target IP-addresses whose ratio of failures/successes is less than 1/10.

**Table 1. A cross-section of the "event" (successful ping group reported back to DPP web-service) dataset.**

| src_ip | dest_ip | time | ping_cnt | ping_avg | ping_stddev |
|---|---|---|---|---|---|
| 196.211.28.141 | 61.190.88.150 | 1269784433 | 4 | 4 | 0 |
| 81.49.2.150 | 41.209.115.174 | 1269784403 | 5 | 139 | 3 |
| 217.159.250.4 | 61.190.88.150 | 1269784406 | 4 | 211 | 217 |
| 217.210.105.214 | 61.50.172.152 | 1269784430 | 5 | 1812 | 874 |
| 87.219.234.4 | 87.5.148.142 | 1269716467 | 5 | 1196 | 952 |
| 86.174.205.122 | 76.251.217.34 | 1269716468 | 5 | 953 | 502 |
| 81.49.2.150 | 88.22.102.231 | 1269784401 | 5 | 146 | 65 |
| 81.234.235.145 | 81.240.60.180 | 1269784408 | 4 | 2209 | 0 |
| 193.52.216.130 | 220.255.0.50 | 1269784403 | 5 | 516 | 281 |
| 81.49.2.150 | 87.242.27.71 | 1269784399 | 5 | 1602 | 2225 |
| 87.65.80.59 | 68.32.252.118 | 1269784405 | 5 | 844 | 439 |
| 77.199.36.8 | 85.13.98.11 | 1269716458 | 5 | 1439 | 341 |
| 217.159.250.4 | 83.77.79.222 | 1269784400 | 5 | 2293 | 0 |
| 77.199.32.55 | 193.151.116.197 | 1269716454 | 5 | 1018 | 1084 |
| 193.252.13.103 | 77.199.32.55 | 1269716451 | 5 | 80 | 0 |
| 190.204.117.27 | 77.199.32.55 | 1269716477 | 5 | 764 | 484 |
| 193.252.13.103 | 24.118.77.216 | 1269716459 | 5 | 1656 | 1058 |
| 77.167.46.13 | 193.151.116.197 | 1269716467 | 4 | 712 | 31 |
| 85.69.9.229 | 85.13.98.11 | 1269716465 | 4 | 1225 | 967 |
| 77.199.32.55 | 88.113.109.232 | 1269716466 | 4 | 1449 | 1155 |
| 76.251.217.34 | 93.41.179.54 | 1269716475 | 5 | 2602 | 171 |
| 193.151.116.197 | 85.69.9.229 | 1269716458 | 5 | 1017 | 0 |
| 193.151.116.197 | 91.210.195.86 | 1269716473 | 5 | 2214 | 554 |
| 24.150.234.55 | 88.113.109.232 | 1269716476 | 4 | 1212 | 1648 |
| 193.52.216.130 | 82.95.139.184 | 1269784397 | 5 | 212 | 238 |

## 3. CONCLUSIONS

The distributed data collection method used in DPP proved to be effective for gathering large quantities of data on the latency time between geographically-distributed clients. Although browser-based pings have an added overhead compared to traditional command-line ping requests, initial observations suggest that they are representative of latency and they are a reasonable approximation to traditional pings.

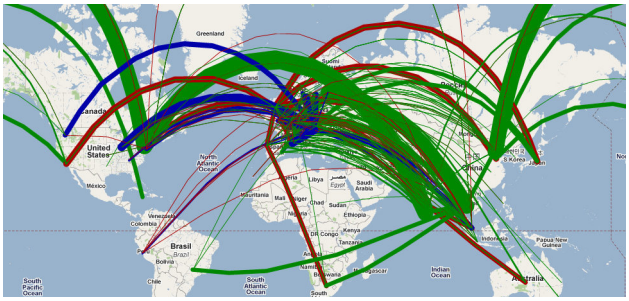


**Figure 3. Small cross-section (~600) of ping events. Increased line thickness indicates longer latency between geocoordinates.**

This paper describes a method for collecting latency data and storing it on a central server. The first significant contribution is the implementation of a browser-based ping algorithm using standard asynchronous javascript. The second contribution is the implementation of a library for performing simple two-way cross-domain communication for passing ping targets and collecting data. What now lies ahead is the interpretation of large latency datasets. As discussed in previous projects (e.g. Wifi survey performed in Hong Kong [1]), the data collected is useful in comparing physical distances with latency times. The next step in the project is to produce visual mashups of connectivity across the globe. Figure 3 shows a straightforward translation of a very small amount of data where lines on the map represent pings and the width of the line is the average ping time. Other data that has been collected (e.g. standard deviation of ping time) is likely to reveal

further information (e.g. reliability and consistency) about the connectivity between clients. Further steps will include a map where the physical location is warped to represent the DPP data.

Finally, as development continues we will be publishing mashup examples, API instructions and source downloads at:

http://www.distributedpings.org/

# 4. REFERENCES


[1] Kwan, White, Tsang, Kwok, Eustace, White, IEEE802.11: WIFI Survey & Visualisation Experiments (Spring 2009), Pearson, Prentice Hall.

[2] Google Analytics, http://www.google.com/analytics