
CS422 - Spring 2008

Lab 2

A PlanetLab Network Coordinate System

Handed Out: March 24, 2008

Due Date: Checkpoint 1 (April 6, 2008); Checkpoint 2 (April 20, 2008) (11:59 pm)

1 Objective

The objective of this lab is to implement a tool which can compute the network coordinates for a set of nodes widely distributed over the Internet. This lab will give you a flavor of the following three popular domains of networking.

- *PlanetLab testbed*: Get a hands-on experience over a real wide-area network testbed, and experience the scale of today's Internet.
- *Network measurements*: Perform measurements on the scale of the Internet, and experience how network traces are collected, sanitized and analyzed.
- *Network algorithmics*: Experience the application of algorithms and some concepts from *physics* to networking.

The rest of the handout is organized as follows:

- *Getting Started*: Get familiar with the Vivaldi algorithm¹ [2], register for an account on PlanetLab testbed, get familiar with PlanetLab testbed, and get familiar with a scripting language.
- *Implement*: Write a tool to measure pairwise network latencies (*ping-times* or round trip times) between all n nodes assigned to your team and build a $n * n$ matrix of these ping-times.
- *Analysis*: Compute the network coordinates using the ping-time matrix by applying the centralized version of Vivaldi and the simple Vivaldi algorithm [2]. Compare the results with respect to convergence time and their accuracy.

2 Getting Started

2.1 Register for PlanetLab account

You will use ssh to access a PlanetLab node. For doing a ssh, you need to set up ssh and generate a RSA (v2) key to access the PlanetLab nodes. Do not use your CS department key for PlanetLab access.

More information on generating a RSA key pair and uploading the public key to PlanetLab is available on

<http://www.planet-lab.org/doc/guides/user>

¹a simple algorithm that assigns synthetic network coordinates to hosts such that the distance between their coordinates accurately predicts the communication latency between them.

Note that PlanetLab infrastructure takes some time to distribute the keys, so register as soon as possible. Also note that each one of the team members will need to create a *separate* account on PlanetLab and the name of the PlanetLab slice which you will be using is *purdue_8*. Make sure to use the same login name and email as your CS department. Your site name is Purdue and your role is *user*.

While following the User Guide, you can directly move from section 3.1 to section 4. PlanetLab slice *purdue_8* has already been created for you and nodes have been already assigned to this slice.

Please read the PlanetLab Acceptable Use Policy (AUP), before creating an account at <https://www.planet-lab.org/aup>

2.2 Accessing PlanetLab nodes

Once your account is activated, select a node from the set of nodes assigned to your team and ssh into that node by following the instructions in the user guide section 4.

Note that there is no need to install anything on your PlanetLab slice, most of the things which you will require are already installed. If you really need anything, let me know first and then follow the instructions in section 4.1 of user guide.

2.3 Your PlanetLab working directory

When you access a PlanetLab node, before starting to work on it - create a working directory named *team < number >* in the home directory, after the name of your team. This is the working directory for your team. All your scripts, programs and data should be collected under this directory only.

2.4 Working on PlanetLab nodes

Each team is assigned a set of 40 nodes on PlanetLab, which you will be using for all your measurements and work. Note that you are not going to use these nodes for compiling programs, and storing anything except your measurement results, and your programs (whenever necessary). Also note to minimize the amount of data (measurements) you store on these nodes at any time. At any time, your workspace should not exceed 40 MBytes on a particular node. To ensure this, routinely pull data from PlanetLab nodes to a local storage. Also make sure to test your scripts and programs on department lab machines before executing them on PlanetLab.

You should not access any node other than the ones assigned to you. Violation of any of these rules will be strictly dealt with.

2.5 Familiarize with Vivaldi

Once you are able to access the PlanetLab nodes, read the section 2 (2.1, 2.2, 2.3, 2.4) of [2]. The Centralized and the Simple Vivaldi algorithm explained in section 2.3 and 2.4 are fairly simple to understand. An overview of these algorithms will be also given in the PSOs.

2.6 Scripting skills

While we do not require you to write your programs in any particular language, it is highly recommended to use a scripting language such as Perl or Python. Note again not to use PlanetLab nodes as compile farms, you should compile your programs on department lab machines and then copy them to PlanetLab nodes.

3 Implement - Collecting pairwise ping-times

As explained earlier, your aim is to compute the network coordinates for the PlanetLab nodes assigned to you using the two versions of the Vivaldi algorithm. For this, the first step is to gather the pairwise $n * n$ latency matrix for the n nodes assigned to you ($n = 40$ for each team).

3.1 Network measurements

Write a program, which accepts a list of foreign nodes to ping. When your program is executed on a PlanetLab node, it should go through this list and ping each other node (except itself after small pauses) and record the round-trip time. Your program should also accept a number which denotes the number of times it should repeat this ping cycle for each node in the list. Output of the program should be all ping times and not just the lowest ping time.

3.2 Data collection

Write another program, which coordinates the execution of the first program on all n nodes assigned to you, and gathers data from each site to a base station (which should be a local node). This program will also help you to routinely pull data from all the nodes, and ensuring that you never exceed your storage limit on a PlanetLab node. You can use tools such as *tar*, *pssh* and *pscp* for this purpose.

A nice example of *pssh* and *pscp* is given at

http://www.ics.uci.edu/~lbao/planetlab/uci_planetlab_instruction.htm

3.3 Sanitization and tolerance to failures

Finally, note that your programs should be tolerant to network and control errors. PlanetLab nodes can go down sometimes. Examples of other errors also include lost ping packets, asymmetric ability to ping hosts and failures to login to PlanetLab sites, file transfer failures and timeouts. You may also need to sanitize the data collected before analysis.

4 Data Analysis

4.1 Compute network coordinates

When you have gathered the ping-time matrices from all the nodes, you will analyze this data to compute the network coordinates of all these nodes using the two versions of Vivaldi algorithm - Centralized Vivaldi and Simple Vivaldi.

4.1.1 Centralized Vivaldi

Write a program to compute the network coordinates of all the assigned n nodes using the *Centralized Vivaldi* algorithm, described in section 2.3 of [2].

You can assume $\|x_i - x_j\|$ as the Euclidean distance between the 2-dimensional coordinates x_i and x_j of the two nodes. Here is a simple tweak while you compute the error force on the spring as $L_{ij} - \|x_i - x_j\|$, which you have to figure out. You can start with all nodes near (but not on) the origin, with initial unit direction vector $u(0)$ in a randomly chosen direction.

Also note that you will use the lowest ping-time between any two machines for the analysis for this part. You should repeat the process described in Figure 1 of [2] until the system converges to coordinates that predicts the error well. Choose appropriate values of *error tolerance* and *timestep*, and make suitable implementation decisions and logical assumptions.

4.1.2 Simple Vivaldi

The centralized Vivaldi algorithm computes the coordinates of all the nodes when all pairwise ping-times are given. This is not scalable for large networks. So the approach described in section 2.4 of [2], solves this by choosing a handful of anchor nodes (say the set A_i) for a node i . Now the node i will execute the Simple Vivaldi algorithm whenever its ping-time to any of the nodes in A_i changes or the coordinates of any nodes in A_i change.

Simulate this algorithm by supplying random samples of ping-times from all the ping-times you have in your data. You should again start with all nodes near (but not on) the origin and random initial unit direction vectors $u(0)$. Make logical assumptions and choices for timestep and tolerance.

Note that in both of the two approaches, you are only simulating the two algorithms by first collecting the ping-times to a base station and then doing all the processing.

Repeat the last two experiments, but use the latitude and longitudes of the nodes as their initial coordinates this time. Latitude and longitudes for all the nodes are already provided to each team.

4.2 Comparison

Compare the results for all the four experiments, with respect to convergence times and accuracy (squared error metric) attained after the coordinates have been computed.

4.3 Map to geographic coordinates

Once you have the synthetic network coordinates for all the nodes, propose a way to map them back to the geographic coordinates.

Write a program, which will take input a node's IP address which belongs to your set of nodes. The program will use the ping-times from the rest of the nodes to this oblivious node to compute its network coordinates. This program will re-use the ping-time data you have already collected.

Now use the mapping program to get the geographic coordinates of this node. Observe how accurately you can predict the geographic coordinates by comparing it with the actual latitude and longitude of this node.

4.4 Questions

I am sure, you would have come up with several questions during the course of this lab, made several assumptions and implementation decisions. The aim of this lab is to make you think in different directions and critically analyze what you implement.

Think about the scalability of centralized Vivaldi and simple Vivaldi algorithms. Think about the utility of simple but elegant tools such as *pssh*, *pscp*, etc. which help the research community to build upon. Think of the power of the most basic tools like *ping* which you were aware of since long.

Hope this lab gave you a flavor of a blend of wide-area experiments on PlanetLab, network measurements and applications of algorithms and concepts from *physics* to the domain of networking - well known as *inter-disciplinary research*.

5 Submission and Grading

5.1 What to Submit

Your submission should contain a directory called **Plab** containing:

- All source files - scripts, programs used to analyze the data-set.

- Your collected data-set (*compressed*).
- A file named METHODOLOGY.txt describing your collection methodology, your assumptions and implementation decisions.
- A README briefly describing how to execute your programs for analysis over the data-set and measurements.
- Your analysis for sections 4.1, 4.2 and 4.3 in a file named *Analysis-Plab.txt*.

5.2 How to submit

1. Make sure you login on to a sslab machine.
2. In the parent directory of your submission directory Plab, type the command:

```
turnin -c cs422 -p plab <submission-dir-name>
```

where <submission-dir-name> is your directory containing your submission (*Plab*).

3. Verify your submission by typing the following command:

```
turnin -v -c cs422 -p plab
```

Do not forget the `-v` above, as otherwise your earlier submission will be erased (it is overwritten by a blank submission).

Note that resubmitting overwrites any earlier submission and erases any record of the date/time of any such earlier submission.

5.3 Grading Criteria

- Your Methodology
- Correctness of output - accuracy and analysis.

*Any concerns about the assignment should be raised by posting on the **newsgroup**, sending email to **msaxena@cs.purdue.edu**, or by speaking to the TA during his PSO sessions or office hours.*

6 Acknowledgement

Thanks to [1] for the general idea for this lab and to John Tim Korb who provided us the PlanetLab slice for developing this lab as part of CS422 for the first time.

References

- [1] Brown university cs296. <http://www.cs.brown.edu/courses/cs296-2/planetlab-ps.html>.
- [2] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proc. of SIGCOMM*, Portland, Oregon, USA, August 2004.