# TULIP Analysis

We use the tulip database to generate our sites.xml which is used in probing the landmarks. We added pingER nodes from the Nodedetails database to the tulip database but with some defined rules. We only added those nodes which contain a traceroute server. To implement this, we developed three packages. Many of the scripts are executed automatically from trscrontab running under pinger@pinger.slac.stanford.edu (see section below).

- TULIP/ANALYSIS/NODEDETAILNODES.pm
- insert_sites-xml.pl
- create_sites-xml.pl

We also add nodes from PlanetLab.

### TULIP/ANALYSIS/NODEDETAILNODES.pm

It is found at /afs/slac/package/pinger/tulip/TULIP/ANALYSIS/NODEDETAILNODES.pm. In order to build this module, we used some predefined perl modules and scripts (require '/afs/slac/package/pinger/nodes.cf' and require Text::CSV_XS). To get data from the **PingER** Nodedetails **database** we used require '/afs/slac/package/netmon/pinger/nodes.cf' and we used standard package Text::CSV_XS to convert our data in comma-separated values. In order to define our node to be a candidate for a tulip landmark we tested it for a few conditions which include that the node must have traceroute server, it should not be set to NOT-SET and its project type should not be set to "D" which means disabled as per Nodedetails database semantics. Nodes which qualified these conditions were put into a separate array. This array is used by insert_sites-xml.pl to insert these sites into the Tulip database

### insert_sites-xml.pl

This perl script is used to create the insert query from the data of the above nodes. Again using theperlpackage Text::CSV_XSwe divide our data into separate chunks. The data is then fed to the structure which contains parameters for the query. This script resolves each host with hostnames taken from NODEDETAILS.pm this helps in eliminating bad hosts if they exist. Before inserting new nodes into the database it checks whether the node is in Nodedetail or not. We use ipv4Addr as our unique key. We traverse through the tulip database and check if there exists some node with the same ipv4addr. If it exists we ignore the entry and if not we go ahead with inserting it in the database. It has a use TULIP::ANALYSIS::NODEDETAILNODES;

### create_sites-xml.pl

This perl script is used to create the sites.xml file which is further used in our TULIP project as a source for node information and landmarks. This perl module uses the template library in order to generate the required xml. It traverses through the tulip database and gets each node, checks the service type and generates the file with all the available parameters in the database. It has a require '/afs/slac/package/pinger/tulip/insert_sites-xml.pl'.

## Tulip Transition to Sites.XML

Our next step in the process is to transform TULIP so that it can get data from created sites.xml. TULIP version 1 was having two different data sources one to get data from nodedetail database and the other was to fetch data from a list containing the Planet lab sites.

In order to perform this step we used JAVA Xerces API, which is described as the best option in terms of resources used and efficiency in parsing XML data. We used the SAX model because of the following reasons.

1) The list of landmarks would increase as more pinger monitoring sites are added

2) We do not require to rewrite or add xml to the document from TULIP

Tutorial posted by SUN is helpful in understanding the parsing of XML. The tutorial can be found here

## TULIP Client

TULIP has a Java based client which has the important functions and classes for sending the query over the web to reflector.cgi. In addition to sending query and getting/parsing responses the TULIP client uses the algorithm(Trilateration or Multilateration) to find the location of queried host. This section explains the technical details about the tulip client and the classes used and important function. The re-implementation of TULIP has led to the use of ANT package management tool by Apache for code compilation.

### Dependencies

Current implementation of the TULIP Java client depends on two files for its execution

- Sites.xml
- /afs/slac/package/pinger/tulip/Initial.txt

The former is discussed above in this article. The later is used to Automate test. Its a list of nodes/sites which TULIP tries to locate when it runs from the command line.

### Code Deployment at SLAC

```
TULIP is deployed at SLAC in /afs/slac/package/pinger/tulip/src
```

The class hierarchy is defined in three folders which are defined in library and tulip client hierarchy.

- Library Folder
  - The lib directory contains third-party packages used for implementation of TULIP. E.g we are using  Java Matrix Class which provides the fundamental operations of numerical linear algebra.
- TULIP Folder
  - This folder contains the actual implementation of the TULIP client. It has two branches core and utill. The core package contains core classes. The utill package contains all the utility classes for the core classes i.e they help in implementation. For instance to get the data we need to parse sites.xml. So, getData is core class whereas the xml parsing class is helping it to achieve its goal so it's in the utill package.

### TULIP Core Classes

TULIP core classes are divided into three sections.

- Automate.java
  - This class contains the control of the program. This class has the "main" function which initiates the test and decides the flow of the program.
- GetPingDataPL.java
  - This class handles all the issues related to parsing sites.xml, querying reflector.cgi and parsing the results. It uses Sites.java as its child class which is in the utill package. Sites.java contains all the parameters like location information, node IP address and rtt values. To populate the geo-location information it parses sites.xml and for the rest of the parameters it parses responses from the reflector.
- Locate.java and PhysicalDistance.java
  - These classes contains the algorithm it self. Once GetPingDataPL has fiished its functionality Automate.java handles the control to these two classes and they use Triangulation/Multilateration Algorithm to identify the location of the host.

## Compiling and Running the Code:

The code has been reformated in a way that it should work with a well know package management tool (ANT). The main benefit of ant is that you do not need to compile a class and set the path once you have edited the class. The package management tool handles it all. The build.xml file has been written to set the paths of classes once the code gets compiled. It would require the following steps to compile and run.

```
cd /afs/slac/package/pinger/tulip

ant -f build.xml
```

This command would compile the code and create a directory build in the same folder with all the .class files. To run the code you need to run the following command ***from build directory*** is needed.

### Running code for List of hosts

There are different command line options available for running the code. Running the code without any options would yield basic test for a list of host in Initial.txt available at tulip directory. The output would be entries in Result.txt which is in /afs/slac/package/pinger/tulip with best three nodes and RTT's.  The format of Result.txt is Target node followed by first best node its RTT then second and third best in the same manner.

```
Run following command from /afs/slac/package/pinger/tulip/build if Class path is not set
 java -cp commons-httpclient-3.1.jar:commons-logging-1.1.1.jar:commons-codec-1.3.jar:. tulip.core.AutomateTest
```

### Running Code for specific Target

The second option available for TULIP command line is running the code for specified target we are trying to locate. The result would be the last line in Result.txt file with the above-mentioned format. Following command is required to run the code to locate that specified target.

```
java -cp commons-httpclient-3.1.jar:commons-logging-1.1.1.jar:commons-codec-1.3.jar:. tulip.core.AutomateTest
target 128.2.115.21 startNewTest
```

### Running Code for List of Targets and updated Log File

If you have conducted new tests and want to run the code on an updated log file, you need to download the log file. This implementation is in place to save the network traffic as code as to go back and forth on the web page and then search for the required targets. In this implementation, we download the log file after every update of log i.e. every time we conduct new tests. The web front end of the log file is at

```
http://www-wanmon/cgi-wrap/reflector.cgi?function=log
```

The command for running the code for a list of targets and required tiers is:

```
java -cp commons-httpclient-3.1.jar:commons-logging-1.1.1.jar:commons-codec-1.3.jar:.
tulip.core.AutomateTest tierFile /afs/slac/package/pinger/tulip/Tiers.txt
targetFile /afs/slac/package/netmon/tulip/Initial.txt downloadLog analyzeLog
```

The output of this command would be a file rtt.txt at $tulip directory. This file would contain rtts of our selected nodes.

Note that the log file by default is on wanmon.slac.stanford.edu in /tmp/tulip_log, e.g.

```
cottrell@wanmon ~]$ ls -l /tmp/tulip_log
s-rw-rw-rw- 1 apache apache 2072902 Nov  1 02:30 /tmp/tulip_log
```

## Selection of Planet Lab Sites

To select the Planet Lab sites we need to go through a process. In this section, we highlight the scripts used in the process and their functionalities.

### generatePL.pl

The process starts with a script /afs/slac/package/pinger/tulip/generatePL.pl
This script generate Sites-yyyy-mm-dd files in /afs/slac/package/pinger/tulip/sitesxml with the nodes which are marked up by the scriptroute administrators. analyzePL.pl. The contents appear as:

```
#Sun Nov  4 03:35:03 2018 /afs/slac/package/pinger/tulip/generatePL.pl: created 360 lines (excluding this line)
on pinger.slac.stanford.edu for pinger from wget --tries=5 --timeout=30 --wait=120 --no-verbose --quiet
'http://www.scriptroute.org:3967/' -O -
Cleveland,"United States",129.22.150.78,41.5074,-81.6053,,NorthAmerica
Vancouver,Canada,142.103.2.2,49.25,-123.1333,,NorthAmerica
Warsaw,Poland,194.29.178.14,52.25,21,,Europe
,"United States",198.133.224.147,38,-97,,NorthAmerica
...
```

#### analyzePL.pl

This script opens up each file generated by generatePL.pl and counts the number of times each node appears in the files. This way we can have the idea for the uptime of every node. And based on the results we can choose the node to be added to our repository(tulip database) as the preliminary passed node.

After primary evaluation, we run tulip from the command line and test the added nodes for their ping servers. The complete test contains 107 sites to be traced by TULIP client.  This generates the log file with all the successful and failing landmarks.

### tulip-log-analyze.pl

The purpose of this script is to parse the log file and provide us a bird's eye view of the landmarks which are failing or are successful. We then disable all those nodes which do not reply to requests by reflector.cgi. This script also generates the total delay/time taken by each landmark to respond. We will use this parameter for the final selection of our tier 0 landmarks.

## Cronjobs

The following scripts are called by trscrontab  to manage and update the tulip data

######################## TULIP ############################################################
#Exercise reflector.cgi so as to update the logs, these should run before tulip-tuning2.pl and landmark-laundering.pl
pinger;120 05 1 * * * /afs/slac/package/pinger/tulip/reflector.pl
pinger 50 15 * * * /afs/slac/package/pinger/tulip/pakcall.pl  # This script just calls the South Asia landmarks to ping slac. It helps laundering.
pinger;100 50 15 * * * /afs/slac/package/pinger/tulip/vtracefromchk.pl

#Update TULIP sites.xml from the NODEDETAILS & TULIP databases
# See https://confluence.slac.stanford.edu/display/IEPM/TULIP+Analysis
# Creating /afs/slac/www/comp/net/wan-mon/tulip/sites.xml
# and /afs/slac/www/comp/net/wan-mon/tulip/sites-disabled.xml must be after creating nodes.cf
#Clean up non-responding landmarks, and restore responding landmarks, tulip-tuning.pl takes about 15 minute.
pinger;30 00 2 * * * /afs/slac/package/pinger/tulip/tulip-tuning2.pl -d 2 -a 1 -t 20 2>&1
pinger;30 07 2 * * * /afs/slac/package/pinger/tulip/tulip-tuning2.pl -d 2 -a 0 -t 35 2>&1
pinger 15 2 * * * /afs/slac/package/pinger/tulip/tier0-tuning.pl
pinger 19 2 * * * /afs/slac/package/pinger/tulip/tulip-dup.pl 2>&1
pinger;120 30 2 1 * * /afs/slac.stanford.edu/package/pinger/tulip/psonar_auto.pl 2>&1 #fetch the list of perfsonar nodes from the lookup servers, then find the ones with working ping server and known location

#the following are run after tulip-tuning.pl since it can enable and disable nodes.
#pinger 20 2 * * * /afs/slac/package/pinger/tulip/landmark-laundering.pl 2>&1
pinger 15 2 * * * /afs/slac/package/pinger/tulip/maintainPL.pl #takes ~ 8 mins. Corrects lat/long of TULIP database using GeoIP Tools
pinger 31 2 * * * /afs/slac/package/pinger/tulip/create_sites-xml.pl > /afs/slac/www/comp/net/wan-mon/tulip/sites.xml #Takes 2 seconds, Must be exected on pinger.
pinger 33 2 * * * /afs/slac/package/pinger/tulip/create_sites-xml.pl --ability 0 > /afs/slac/www/comp/net/wan-mon/tulip/sites-disabled.xml #Must be executed on pinger
pinger 35 3 * * * /afs/slac/package/pinger/tulip/generatePL.pl 2>&1 #Takes 3 mins.
##########################################################################
#The following scripts generate xml files for all of pinger (+perfsonar) and planet lab nodes and just the active ones too.
pinger 51 2 * * * /afs/slac/package/pinger/tulip/generatexmlnodes.pl 2>&1 # Must run on pinger
##########################################################################
#Backup Tulip database, Umar needs to look at this, it is not found on pinger-new, must run on pinger
#pinger 00 8 * * * /bin/bash /afs/slac/package/pinger/tulip/backup_tulip_mysql.sh
pinger 00 8 * * * /afs/slac/package/pinger/tulip/backup_tulip_mysql.pl -b /afs/slac/package/pinger/tulip/DB_backup/ #Must run on pinger
#wanmon 04 0,6,12,19 * * * /afs/slac/package/pinger/tulip/tulip-log-analyze.pl >/afs/slac/u/sf/cottrell/tulip_log #wanmon not available to trscrontab on pinger
#There is a commented out trscrontab entry  on www-wanmon. This job analyzes the /tmp/tulip.log file on wanmon from the execution of reflector.cgi (or its wrapper reflector.pl) to provide a summary table of
#ping successes from PingER, perfSONAR and PlanetLab MAs. Output is to STDOUT.

#wanmon 04 0,6,12,19 * * * /afs/slac/package/pinger/tulip/tulip-log-analyze.pl >/afs/slac/u/sf/cottrell/tulip_log #tulip_log is empty