TULIP Landmarks Laundering

After many tests we concluded that Planet Lab landmarks (http://comon.cs.princeton.edu/status/tabulator.cgi?table=table_nodeviewshort&format=nameonly&persite=1&select='resptime>0'), and to a lesser extent the PingER landmarks, are unpredictable in terms of availability. The TULIP Landmarks Map identifies the landmarks that are currently in use and those that have been disabled since they are not responding. If the landmark is not available the TULIP thread has to time out and this can dramatically extend the measurement duration and insert unnecessary traffic on the Internet. Thus, ideally, requests for pings should not be sent to such landmarks, i.e. they need to be removed from the list of active landmarks. The purpose of this web page is to indicate how we accomplish this.

For every access by TULIP the response for each landmark is logged. We analyze (http:www-wanmon.slac.stanford.edu/cgi-wrap/reflector.cgi?function=analyze&days=1) this logging information and generate a list containing nodes with corresponding success or failure percentages and reasons of their failures. These percentages are generated by /afs/slac/package/pinger/tulip/tulip-log-analyze.pl and the results can be seen here. The idea is that landmarks with a low success rate should be removed from the active list of landmarks. This is done by disabling the entry in the tulip database.

There are two ways in which we can disable the landmarks which are not responding.

1. Manual Process
2. Automated Process (updated by trscrontab job running in pinger@pinger.slac.stanford.edu)

Manually disabling hosts

This is accomplished via the tulip data base. In the tulip database, the table landmarks has a parameter "enabled" which is used to decide which landmark is to be added to sites.xml (the list of active landmark sites). This XML file is later used by the reflector to query the active landmarks for the results. Sites.xml is generated by a trscronjob so if we change the value of enabled to '0' it would automatically not appear in Sites.xml after it has re-run through the trscronjob. We can also update Sites.xml manually. The process is discussed below.

- Login to tulip database (username and password available in escrow -c iepm iepmacct)
- Change the database to tulip by cmd

```
mysql> use tulip;
```

- Now update the value of enabled using the following sql cmd; in this instance we are using ipv4Addr = 141.22.213.35; generally ipv4Addr is the primary key but we can also use hostName as an identifier to disable landmarks

```
update landmarks set enabled = '0' where ipv4Addr = '141.22.213.35';
```

- Update Sites.xml so that it can now use the updated landmarks using follwing cmd

```
create_sites-xml.pl > /afs/slac/www/comp/net/wan-mon/tulip/sites.xml
```

Automated Process (updated by trscronjob)

reflector.cgi

The reflector.cgi script is run twice nightly by the trscrontab on pinger@pinger.slac.stanford.edu to ping the target www.slac.stanford.edu. The two runs are actually run by calling reflector.pl. The first call to reflector.cgi is to use the enabled (ability=1) landmarks. It (reflector.cgi) obtains these (enabled landmarks) from the URL http://www.slac.stanford.edu/comp/net/wan-mon/tulip/sites.xml. The second time is to use the disabled landmarks (ability=0). It (reflector.cgi) obtains these (disabled landmarks) from http://www.slac.stanford.edu/comp/net/wan-mon/tulip/sites-disabled.xml. Running it regularly ensures the tulip log files (their location is defined by /afs/slac/www/comp/net/wan-mon/tulip/log.conf and they are kept in /tmp/tulip_log on www-wanmon) are current and therefore the analysis is also current.

Reflector.cgi has to run on a host in the SLAC network (134.79/16) since that is what the PlanetLab cookie allows. Since one cannot remotely run a trscronjob on www-wanmon, there is a script (reflector.pl) to execute reflector.cgi twice (with ability=1 and then ability=0) via a wget command.

Reflector.pl creates two log files /tmp/reflector.log-enabled and /tmp/reflector.log-disabled which can be reviewed. They are stored on the host that ran reflector.pl (usually pinger.slac.stanford.edu)

- To review reflector.log-disabled on pinger@slac.stanford.edu you may want to use

```
>grep transmitted /tmp/reflector.log-disabled
Landmark(2)=http://206.117.37.4:3355, Client=134.79.104.80, target=134.79.18.188,\
ability=0, 10 packets transmitted, 10 received, 0% packet loss, rtt min/avg/max = 9.485/9.6527/10.007<br>
Landmark(2)=http://pinger.cern.ch/cgi-bin/traceroute.pl?target=134.79.18.188&function=ping, Client=134.
79.104.80, ability=0,\
 5 packets transmitted, 5 received, 0% packet loss, rtt min/avg/max = 171.147/171.627/172.808<br>
Landmark(2)=http://204.178.4.164:3355, Client=134.79.104.80, target=134.79.18.188,  ability=0, 10
packets transmitted,\
 10 received, 0% packet loss, rtt min/avg/max = 86.906/106.1068/154.922<br>
```

- To review reflector.log-enabled on pinger@slac/stanford.edu you may want to use

```
>grep failed /tmp/reflector.log-enabled
Landmark(1)=http://192.42.83.252:3355, Client=134.79.104.80, target=134.79.18.188,  ability=1,\
 failed to connect response code 200 <br>
Landmark(1)=http://138.238.250.157:3355, Client=134.79.104.80, target=134.79.18.188,  ability=1,\
 failed to connect response code 200 <br>

>grep transmitted /tmp/reflector.log-enabled
Landmark(2)=http://pinger-ncp.ncp.edu.pk/cgi-bin/traceroute.pl?target=134.79.18.188&function=ping,\
 Client=134.79.104.80,  ability=1, 5 packets transmitted, 5 received, 0% packet loss,\
 rtt min/avg/max = 316.461/316.685/316.896<br>
```

tulip-tuning2.pl

Using the the Tulip log analysis script's results for the last 1 day for enabled landmarks and results for the last 1 days for disabled landmarks we can identify the hosts and their success percentages. We opted to disable all the enabled hosts that were having success less than 20%, and to enable the disabled ones with success rate greater than 35%.

The tulip-tuning2.pl script is in /afs/slac.stanford.edu/package/pinger/tulip. It uses the perl LWP package to call reflector.cgi?function=analyze&days=1&ability=[1|0] to access the Tulip analyzed log data for the last 1 days, downloads the analyzed tulip log file by requesting it from the reflector using http://www-wanmon.slac.stanford.edu/cgi-wrap/reflector.cgi?function=analyze&days=3 and saves it in a file and then parses the output to get: for option ability=1, the faulty landmarks (i.e. the enabled ones with below 20% success rate by default) that are updated in the Tulip database to disable them; or option ability=0 the disabled landmarks with a success rate greater than 20% that are then re-enabled in the database.

It (tulip-tuning2.pl) is run twice nightly (see the trscrontab) once to disable non working landmarks (those enabled landmarks that have fallen below 20% success), once to enable landmarks that are now working again (those disabled landmarks that now have success above 35%).

```
/afs/slac/package/pinger/tulip/tulip-tuning2.pl -d 1 -a disabled
```

It (tulip-tuning2.pl) must be run on a host in the 134.79/16 address space (i.e. a machine on the SLAC 134.79 address space) and is run before the sites.xml or sites-disabled.xml are created by http://www-dev.slac.stanford.edu/cgi-wrap/scriptdoc.pl?name=create_sites-xml.pl.

The output generated by tulip-tuning.pl is placed at /afs/slac.stanford.edu/package/pinger/tulip/tuning_log. This log file, contains blocks of logs (stanzas) for each run until a month has passed, when it truncates the oldest entries. Each block starts with a unix time stamp embedded in hyphens, indicating when tulip-tuning.pl ran, and ends with *_END_*

A copy of the analyzed Tulip log is also saved at /afs/slac/package/pinger/tulip/analyzedump_[enabled|disabled]

You can run

```
tulip-tuning2.pl -d 1 -a disabled --debug 0
```

from the command line to see how it matches the landmarks in the TULIP database with the log to find ones above the threshold and enable them.

tier0-tuning.pl

After Vtrace, it was observed that all working tier0 landmarks were being disabled by tulip-tuning2.pl, this was due to routers that don't respond to pings. According to reflector logs (tulip-log-analyze.pl) these landmarks appeared to be down even though in reality the targets were causing pings to fail.

To overcome this issue tier0-tuning.pl was written which uses only slac.stanford.edu as the target for deciding which tier0 landmarks should be enabled /disabled. Instead of relying on tulip log, this script calls reflector directly and parses the output for decision making. This rules out targets as the cause of ping failures.

The script is at:

/afs/slac/package/pinger/tulip/tier0-tuning.pl

tulip-dup.pl

There are several cases where we have more than one landmark at the same geographic location. Having more than one active landmark at any location just results in additional geolocation time without improving accuracy. This script finds all the enabled landmarks that have the same geographic location and disables all except one. This script runs every night around 2am from trscrontab.

You can run it manually from here:

```
/afs/slac/package/pinger/tulip/tulip-dup.pl
```

After running the above scripts

To generate the sites xml files it is necessary to run:

```
/afs/slac/package/pinger/tulip/create_sites-xml.pl > /afs/slac/www/comp/net/wan-mon/tulip/sites.xml #Takes 2
seconds
/afs/slac/package/pinger/tulip/create_sites-xml.pl --ability 0 > /afs/slac/www/comp/net/wan-mon/tulip/sites-
disabled.xml
```

They take a few seconds to run.
They set up the lists of enabled and disabled landmarks. These are needed by reflector.cgi.
It may also be necessary to run /afs/slac/package/pinger/tulip/generatePL.pl to get the country, region and lat/long of the PlanetLab sites and add them to the Tulip database.

Then run:

```
/afs/slac/package/pinger/tulip/generatexmlnodes.pl
```

To get the landmarks types and ability for the Tulip map. It produces active-rss.xml and disable-rss.xml. These are used by the Tulip map.

Duplicate landmarks

Looking at the enabled landmarks (.e.g. http://www-wanmon.slac.stanford.edu/cgi-wrap/reflector.cgi?function=landmarks) and copying and pasting into Excel then sorting on latitude, it is apparent that (especially for PlanetLab landmarks) there are multiple enabled landmarks at single sites. For example:

swsat1502.mpi-sws.mpg.de
swsat1503.mpi-sws.mpg.de
swsat1505.mpi-sws.mpg.de,

One could look at the analyzed log (see http://www-wanmon.slac.stanford.edu/cgi-wrap/reflector.cgi?function=analyze&days=2) for the success of the landmarks to select the best for each site and disable the others. This might speed up TULIP quite a lot for Europe and N. America.

I created an Excel spreadsheet to compare the above 2 files after a bit of clean up of the analyzed log. If we remove such redundant landmarks, then that removes 129 landmarks out of about 360 landmarks. Even if TULIP can't recognize the region it might 30% faster. It should also be similarly faster for targets in Europe and N. America where most of the PlanetLab landmarks exist.

The spreadsheet indicates the duplicate landmarks that can be disabled in a red background

Site Contacts/Owners

perfSONAR sites

As a development effort the perfSONAR developers seldom contact owners of deployed perfSONAR machines. Thus we reach out to them directly representing SLAC/PingER (i.e. not representing the developers, and without a to CC). We note we are researchers that want access to the measurements, and things aren't working. You can find the contact information for a given note on it's home page, or you can use the LS infrastructure.

PingER sites

The site owner can be found in the NODEDETAILS database.

Access to Tulip MySQL DataBase

You need to load the MySQL Workbench. Make sure you are on VPN or connected to the internal SLAC Ethernet. Logon as user tulip, the password is given in escrow edit -c iepm iepmacct. Open tables ->select a table, e.g.landmarks -> right click and Select Rows. Go to File menu ->Open MySQL Script -> default SQL. Then you can edit the SQL command move the mouse over to highlight the command and then click on the lightning icon at the top left.