Design Chapter

Andrew van Rooyen

1. DESIGN AIMS

The system will provide mechanisms to compare how the following protocols handle file transfers. This information will be used to determine which one is best to use when dealing with bioinformatics data on an educational network.

— GridFTP
— FTP
— HPN-SSH
— SCP

The protocols to test were specified by the proposer of the project. They are also the most popular protocols in the field, perhaps with the exception of *fasp* by AsperaSoft [Beloslyudtsev 2014], which is non-free. Because of the popularity of these protocols over all others, it is assumed that one of them will be the ideal protocol.

2. APPROACH

Transfers using each of the 4 protocols above will be run while the network traffic is logged. This information will then be analysed.

A simple python script will be sufficient to run the transfers, as most of the work will be done by calling a subprocess for each specific protocol. The analysis will also be done using python as there are a vast number of analytical and visualisation tools available.

The testing environment will be kept as stable as possible during tests, and multiple tests will be run at different times of the day.

First, the scripts which run the transfers will be written so that tests can be run and results saved. The analysis suite will be written later, and will accept the previously-saved results as input.

The testbed will be constructed using git for version control. The repository is available at https://github.com/wraithy/bigbinf.

With regards to GridFTP, the 'lite' version will be used. This means that authentication is done over ssh as opposed a previously-configured certificate authority. This does not make any difference to the file transfer itself, but it does prevent unnecessary configuration of the testbed which can be quite complex in the case of 'full' GridFTP.

Once the ideal protocol has been decided on, it will be made available as an endpoint to the microcloud system, so that users can retrieve their results in an optimal way.

3. EVALUATION

The system will log packets on the network interface used for the transfer. This allows for analysis which goes deeper then simply logging the transfer speed and file size

reported by the program.
Metrics that will be investigated include

— Average speed, max speed
— Consistency
— Total transferred bytes and overhead

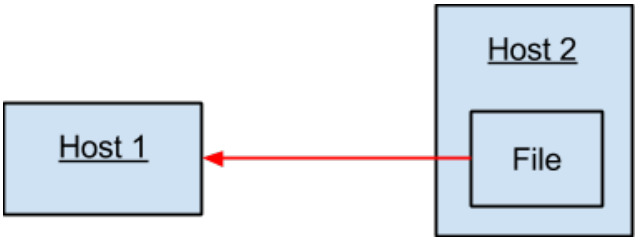
4. IMPLEMENTATION

All testing will be done on an Ubuntu 14.04 system, with the following packages installed

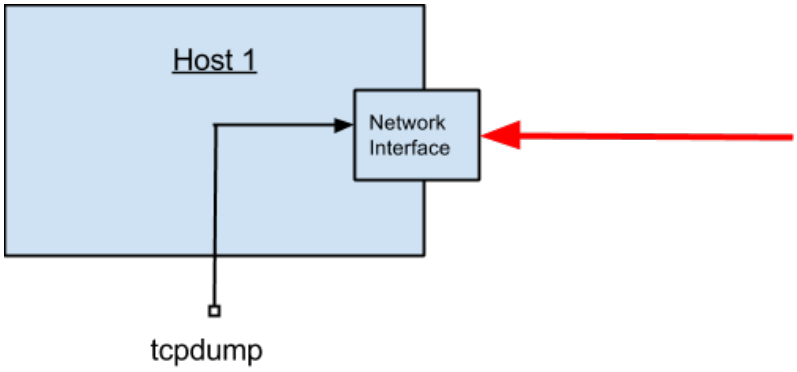
— python 2.7
— globus-gass-copy-progs 8.6
— globus-gridftp-server-progs 6.38
— tcpdump
— openssh-client
— openssh-server

However, the testbed should work on any Unix system as long as python, tcpdump and the correct programs for each protocol are installed.

4.1. File transfer



A copy is initiated from Host 1. A file from Host 2 is transferred to Host 1.



The ‘tcpdump’ program (www.tcpdump.org) comes with most Unix systems. It watches a network interface (e.g eth0) and logs information about packets which pass through. The program is run while the copy is in progress, and the output is filtered to include only packets sent between Host 1 and Host 2.

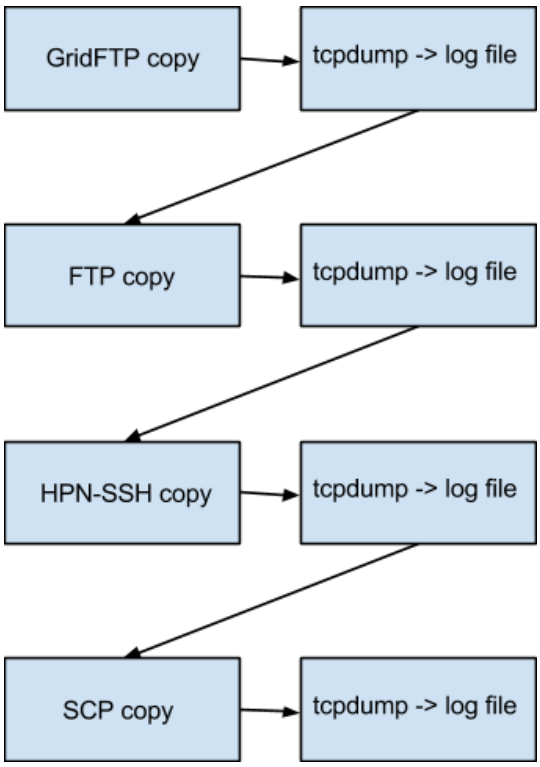
Note that despite the name, tcpdump can also capture UDP packets, which is relevant for some of the protocols.

4.2. Transfer logging

The python script for running the file transfers has been written to accept

— the name of the network interface
— the remote hostname (Host 2)
— the path of the file on Host 2
— a local path to copy the file to

It then resolves the IPs of each host, and for each protocol, runs a copy in isolation. It spawns a tcpdump subprocess, and lets it run for precisely as long as the copy runs. It then saves the output in a file.



Doing it this way allows for a much more controlled environment. Because the tcpdump is only capturing while the copy is running, no other packets will be included in the logs. Also, the copies are run programmatically and consecutively. Following copies are not started until both the tcpdump and protocol processes have been closed, and the log file has been written. This means that they are all run in an identical (within reason) environment, but at the same time do not interfere with each other.

This test process is run multiple times for statistical reasons, generating multiple log files.

4.3. Analysis of dumps

A separate python file reads in the log files and parses them. Operations can then be run by looking at the time of each packet, and the size of its payload. This allows analysis of

— overhead, because *all* transferred is logged, rather than just the file size
— speed (data/time)
— consistency
— total time
— total size

This data can then be aggregated over multiple log files, and graphed using the matplotlib python library.

More info is needed here, and will be filled in once I have completed the analysis scripts.

REFERENCES

Dima Beloslyudtsev. 2014. Aspera transfer guide. (2014).