P2P network Simulation kit

User Guide

-

Alan Davoust

-

December 2010

This is a brief user guide for the P2P network simulation tools found near this document.

The tools consist of the following components

- a collection of DEVS models implemented using the CD++ that can be used to simulate the interaction of peers in a peer-to-peer file-sharing network;
- a Python script to generate parts of the DEVS models for (possibly) large numbers of peers;
- a Python script to pre-process the log files resulting from a CD++ simulation, with the purpose of viewing a simulation run in the next tool...
- a java Applet to view a simulation run as it happened, in an animated graph.

1 Requirements

CD++
In order to run simulations, it is assumed that CD++ is installed in eclipse, with the workspace directory "\eclipse\workspace".
If the eclipse workspace directory is different, the testing scripts (*.bat files) will have to be slightly modified as they explicitly refer to this directory.
To the publicly available CD++ distribution, the interested user can also add the latest version of `distri.cpp`, that we have updated : it seems that the seeding of the random number generators were not working properly. They may have relied on standard libraries that have now changed... alternatively, the authors may be using outdated standard libraries.... in any case if the users find that their random numbers follow alwayts the same sequence, then consider using the distribution class provided here. It should be copied to eclipse/plugins/CD++ builder / internal.

Disk Space
Some of the simulations tend to generate large log files. 100Mb is not impossible... be aware.

Python
A python interpreter is required to run the python scripts. In addition, the numpy library is required by one of the scripts. Numpy can be obtained for free from the address :
http://sourceforge.net/projects/numpy/files/

Without the numpy library, the scripts to generate new file-sharing networks will not work. However, an example network, with 90 peers, usable as is, is provided.

Java
The java applet is provided as a runnable jar file. However, it does require a Java Virtual Machine supporting java 1.5 or later.

2 Simulating a Peer-To-Peer file-sharing network

Running and viewing a file-sharing network can be done with the following steps.

1 Create a network model
You can create a new network model with your chosen number of peers, randomly acquainted to one another.
This step is optional, as an example network with 90 peers is provided : see the directory `p2p/coupled/Network90`.
To create your own network, simply run the python script `graphgenerator.py`.
You will be prompted for the nuymber of nodes you want in your network. At this point, that is the only customizable aspect of this script. The script will generate all the necessary files for the model, and place them in a directory `Network`*XX* where *XX* is the number of nodes that you chose.

2 Compile your model
Within eclipse, use the "build" button in the CD++ perspective, or use the make file.
This step is mandatory : the provided model with 90 nodes is not compiled.

3 Run the simulation
Among other files, the python script invoked in Step 1 has created a handy script to directly run the simulation. Look for it in your `NetworkXX` directory: the file is called `P2PTest.bat`.
Run that windows batch file, either by double-clicking or calling it from the command line. Some of our models are somewhat verbose, and so if you are testing a network with many peer (say, more than 20) we recommend diverting the output to a file. Example:
```
C:\eclipse\workspace\p2p\coupled\Network90\>P2PTest.bat > results.txt
```
Note that if your workspace directory is different, the script will not work as such. You can simply delete the first line (`"cd..."`) and invoke it from the command line, or use the "simulate" button in eclipse.
However, we found that clicking the bat file directly from eclipse started the command line with the working directory set to the desktop. For this reason the bat file includes a change directory command.
Note that the file of interest to you for visualizing the simulation results will in fact be the log file – `P2PXXLog.log` if you used the script, where XX is, as before, the number of peers in the network.

4 Preprocess the log file
The above log file (`P2PXXLog.log`) can be used for visualizing the simulation run in our java applet, but it requires a little pre-processing. This pre-processing is done with the other Python script provided, `LogFileReader.py`.
This pre-processing step can be skipped as we provide a sample pre-processed log file, `ProcessedLog.txt`.
To process a file, just run the script `LogFileReader.py`. You will be prompted for the name of the file to process. The output will go to afile named `ProcessedLog.txt`.

5 View the simulation results.

This can be done with the provided java applet, found in the archive `RunnableP2PViewer.jar`.
To run it, simply double-click on the jar file, or invoke it from the command line :

```
>java -jar RunnableP2PViewer.jar
```

A user interface will pop up and provide you with some instructions.
Select the file that contains your simulation data : it must be in the format of a processed log file. The provided `ProcessedLog.txt` is an example of a usable file.
The file will take a short while to load, and then another screen will pop up and you will see a graph with red and blue nodes floating all over the screen. The layout algorithm for the graph is still being computed.
The red circles represent peers, and the smaller red squares are documents stored by the peers. This graph shows you all the peers and documents that will be represented during the simulation. This layout, once finalized, is then reused in an animation where you see the peers go online, connect to one another, make and answer queries, and publish documents.
The animation starts after you have stopped the layout algorithm by clicking on the button at the bottom of the screen.
Until then you can also help it out by moving the nodes around yourself : select "PICKING" from the "Mouse Mode" drop down box at the bottom, and just select any node: you can drag it across the screen to a better location. Note that the algorithm will continue to process the layout and the node may not stay where you moved it to. For best results, select a peer (a red circle) and all the documents that are attached to it (the smaller square nodes).
In "TRANSFORMING" mouse mode you can zoom and move the whole layout around so that it comfortably fits into your visualization screen.
Note that both mouse modes will still function while the animation is running.
During the animation, you will see peers appear and disappear, connect, publish documents : this is simply reflected by the changes in the graph structure.
Peers also output queries : a peer that outputs a query changes color and becomes purple for approximately 1.5 seconds, then goes back to normal.
A peer than answers a query changes color to pink, for only half a second. When the peer goes pink, the document that the peer is reporting as a query answer also flickers pink.
In the following seconds you should see a copy of the document appear next to the peer that was making the query.

Note the "fast-forward" button. It can be used to toggle between normal speed and fast-forward mode.
At "normal speed", the simulation is viewed following the timeline of the simulation : if a simulated event is supposed to happen after 17 minutes, you will see it after 17 minutes of animation.
In "fast-forward" mode, the events are all shown in rapid sequence, with only 50 milliseconds interval, whatever their simulated timeline was.
Additional commands (pause, reverse...) are being prepared for future versions of this applet.
If the user has problems with this applet, an online version can be viewed at the following address: http://sce.carleton.ca/~adavoust/
(see the link "P2P simulation demo" on the left)

Thanks you for using our P2P simulation tools.
The java code for the applet is found inside the runnable jar file : open it with an archiver such as WinRar, and see the directory "src".