Peer-to-Peer Simulators

Mark Baker and Rahim Lakhoo

ACET
University of Reading

{Mark.Baker@computer.org, r.n.lakhoo@rdg.ac.uk}

Date: 1st May 2007

Abstract

In this technical report, we present our findings on network simulators, which can be used to help test and design a P2P system. The Portal-based P2P system under development uses portals to provide a user interface and P2P volunteers to provide resources to the network. The aim is to create an infrastructure that can be used by the scientific community, based on existing social networks, to help download and maintain large datasets. Network simulators provide a virtual network environment, which can potentially provide assurance for software developers, when making design decisions. This report outlines the architecture of the Portal-based P2P system and defines a set of criteria for a suitable simulator, to aid the development of the P2P system. A number of network simulators are reviewed and briefly tested for their suitability based on criteria. The experience gained from reviewing network simulators is presented in this report.

1. INTRODUCTION 3

1.1 P2P SYSTEMS 3
1.2 PORTAL-BASED P2P SYSTEM 3
1.3 NETWORK SIMULATORS 5

2. REVIEW 5

2.1 CRITERIA 6
2.2 SIMULATORS 6
2.2.1 NS-2 and NAM 6
2.2.2 PeerSim 8
2.2.3 PlanetSim 8
2.2.4 OMNeT++ 8
2.2.5 OverSim 9
2.2.6 GPS 9
2.2.7 AgentJ 10
2.2.8 P2PSim 10

3. INSTALLATIONS AND TESTS 11

3.1 NS-2 AND NAM 11
3.2 PEERSIM 12
3.3 PLANETSIM 13
3.4 OMNET++ 13
3.5 OVERSIM 15
3.6 GPS 16
3.7 AGENTJ 17
3.8 P2PSIM 17

4. SUMMARY AND CONCLUSIONS 18

4.1 SUMMARY 18
4.2 CONCLUSION 20

REFERENCES 22

1. Introduction

In recent years, Peer-to-Peer (P2P) technologies have become increasingly popular. A P2P system can be defined as a distributed network architecture, whereby participants share a part of their own hardware resources, such as processing power, storage capacity, or network bandwidth. The shared resources are necessary to provide the service and content offered by the network, such as file-sharing. The service or content provided by the P2P network is accessible by other peers directly, without passing intermediary entities [1].

1.1 P2P Systems

P2P systems are commonly used to distribute content as part of a Content Distribution Network (CDN). One of the main reasons for the success of P2P networks is providing users with the ability to distribute content by contributing bandwidth back to the network. This concept is the social aspect of a P2P network, whereby users freely give bandwidth to other users. Most, if not all, members of a P2P network contribute resources to the network; essentially users are helping each other download files. Popular examples of P2P networks/applications used are Gnutella [1], BitTorrent [3], Kazaa [4], eDonkey [5], Limewire [6] and Freenet [7].

P2P technologies are mainly used for file sharing and are becoming the normal method for the distribution of Linux operating systems. The P2P method of file distribution is superseding the older mechanisms, such as the File Transfer Protocol (FTP). When using traditional FTP servers, the bandwidth for all users downloading from the server is split between the numbers of users. This can lead to a server's bandwidth being completely depleted, leaving users with a less than optimal transfer rate. The centralised architecture of the traditional FTP server is known not to scale well. However, P2P systems such as BitTorrent have been seen to scale to over 100,000 users for a single file/data set. P2P applications like BitTorrent, which have made efficient use of bandwidth have forced Internet Service Providers (ISP) to reassess their price plans [8], due to the increased flow of traffic from broadband users. Even though broadband users do not have much bandwidth, when compared to enterprise Internet connections, they still manage to accumulate enough bandwidth to transfer many Gbytes of data. A study monitored over 90 thousand BitTorrent users, downloading a file of approximately 2Gbytes, over a period of 8 months. During June 2004, BitTorrent was 53 per cent of all P2P traffic on the Internet backbone [9].

1.2 Portal-based P2P System

The aim of the Portal-based P2P system is to allow users to store large datasets across many peers, instead of a dataset being hosted centrally. Each peer in the network contributes resources back to the network. Resources contributed, are network bandwidth and file storage space. Portals within the P2P system provide users a way to manage the dataset hosted by the P2P network. Our Portal-based P2P system is designed to manage and update large scientific datasets. Projects such as the Sloan Digital Sky Survey (SDSS) [10] are set to produce over 15 TBytes of data by the end

of the project lifetime [11]. Such projects have datasets, which are growing, and subsequently the costs and logistics of transferring the datasets is becoming increasingly difficult.

Our P2P system consists of a two-layered peer hierarchy, see Figure 1 for an overview. The peers are arranged in a hierarchy of portal, and volunteer peers. Portal peers host a portal that provides an interface for users of the P2P system to manage the data stored, along with the contribution of resources. Volunteer peers are users who contribute a user-defined amount of resources to the network. The backbone tier (portal peers) consists of a portal service hosted, potentially by, academic institutes or research laboratories, which are part of a particular project. The volunteer peers consist of individuals and/or organisations that are either involved or interested in a particular project or dataset. Volunteers will run a mini-peer within their Web browser that contributes to the distribution of content. All peers in the Portal-based P2P system will provide a distributed service for the network, so that peers can find each other and data to download, this is commonly known as a tracker. The tracker will be implemented using Tycho [12], which provides a virtual registry and an asynchronous messaging framework. Tycho allows consumer/producer network end-points to discover and communicate with each other over a wide-area network.

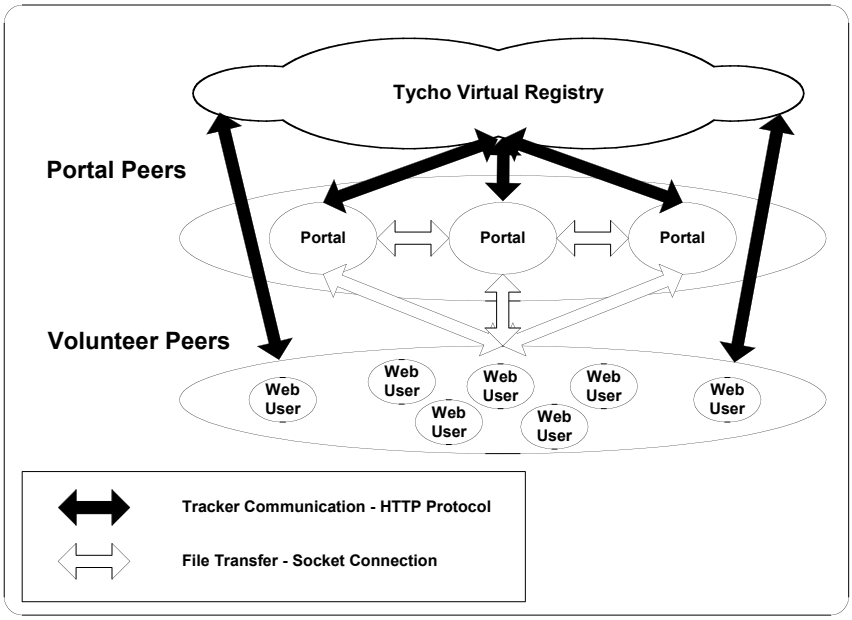


Figure 1, Portal-based P2P Network Overview.

In order to provide greater assurance to the development and testing of the Portal-based P2P system a simulator is needed.

It was decided that before implementing the P2P system, a P2P simulator should be used, to model the Portal-based P2P system. This was to expose unaccounted anomalies with the performance of the algorithms in use to store and distribute the data. A simulator should allow us to experiment with different set-ups and configuration scenarios. In theory, a simulator should help us to test and develop our solution off-line and potentially with greater assurance. Network simulators can reduce the development time for a P2P system. Therefore, it is important to review the various simulator implementations, which provide support for P2P systems.

1.3 Network Simulators

Network simulators are typically used to simulate network communications in particular scenarios or situations, without configuring 'real' machines or networks. Simulators can help with the development and testing of a network application. There are two main types of network simulators, packet-based and flow-based. Packet-based network simulators, attempt to simulate data packets, such as NS-2 [13]. Other simulators are flow-based, and work at the application level, which means they disregard parts of the TCP/network stack. Several flow-based simulators, such as GPS [14] have a mechanism to introduce packet delay, to provide realistic communication characteristics, while others do not. Typically, packet-based simulators take longer to complete a simulation than flow-based simulators. This is because of the calculations made for each packet in the simulated network. Network simulators normally allow a developer to produce a network topology and define delay, bandwidth and connection/traffic characteristics for the nodes and links.

Lately, network simulators have evolved to allow the simulation of P2P systems. Network simulators such as NS-2 have been used for testing P2P protocols, while other network simulators, like OMNeT++ [15] have been forked to produce a simulator specifically designed for P2P systems, namely OverSim [16].

P2P system development has many common issues found with programming other distributed systems, such as difficulty when debugging. P2P systems add further complications, especially when testing different topologies and setting–up scenarios across multiple machines. Finding stable wide-area network connections and suitable machines on these links is also a problem and can be impractical. Because the environment is not fully controlled and results gathered may not be consistent. The time taken to develop a P2P system can be longer, due to these issues. P2P network simulators try to address the increased difficulty when designing or testing P2P systems, by providing a virtual network environment.

This report will discuss the various network simulators suitable for P2P networks and their features. We outline our experiences using the simulators and their ability to model the Portal-based P2P system. Section 2 will review the network simulators and provide the criteria for their suitability for our system. Section 3, discusses issues and configurations with the simulators. Finally, Section 4 summarises and concludes.

2. Review

Simulators can be classified into two categories, packet-based and application-level. Packet-based simulators calculate delay, bandwidth and routing for each packet generated or used by the simulation. Application-level simulators do not account for each packet instead they calculate bandwidth and delay to/from network end-points. Application-level simulators usually use the terms 'flow-based' or 'message-based' to describe how they evaluate communications between nodes in the simulation.

This section will review several network simulators suitable or specifically designed for P2P systems. We define the criteria for the simulators, which lists our needs from a simulator. Following this, we move on to detail the simulators, which includes NS-

2, NAM [17], PeerSim [18], PlanetSim [19], OMNet++, OverSim, GPS, AgentJ [20] and P2PSim [21].

2.1 Criteria

The simulator needs to be able to implement our custom protocols, network topologies and be configurable for different scenarios. Our main criteria for a P2P simulator, is to have the ability to implement our protocol/system in Java, and be able to graph a multi-tier network. The simulators are required to:

- Allow the implementation of custom P2P protocols,
- Provide facilities for multi-tier topologies,
- Provide visualisations, e.g. an overview of the network,
- Provide reasonably accurate results in terms of 'real-world' performance,
- Have good support and/or documentation,
- Have the ability to interface with Java.

Other criteria include the scalability of simulators for simulations with a large number of nodes. Another non-essential feature for a P2P simulator is a distributed simulation. This would allow us to run a large simulation across a cluster. Results or simulator output is also examined at in terms of suitability and usability. Ideally, the simulator should also have a BitTorrent-like protocol, specifically a swarming feature, as our Portal-based P2P system is loosely based around the BitTorrent protocol.

2.2 Simulators

This section will discuss and outline each network simulator, starting with NS-2 and NAM. Each simulator will be compared to our criteria to discover its suitability for our needs.

2.2.1 NS-2 and NAM

NS-2 [13] is a well-established network simulator within the research community, which has been actively developed for 16 years. NS-2 started as a variant of the REAL network simulator [22]. Development was supported by DARPA through the VINT project [23]. NS-2 has support for TCP, routing and multicast protocols over wired and wireless networks. NS-2 uses C++ and an object orientated version of TCL called, OTCL. NS-2 is described as an event-based simulator for packet-level simulations. The simulations are written in TCL scripts. The TCL scripts define the nodes and the characteristics of the communications links, while protocols are implemented in C++.

NS-2 does not include any visualisation tools by default. NAM is a network animator developed in TCL/TK to provide visualisations for the NS-2 simulator. NAM provides packet-level animations for the simulations. NAM also allows users to arrange network graphs to aid designing and debugging of network protocols.

GT-ITM [24] is used to create topologies of internetworks. GT-ITM can be used to generate topologies for NS-2. GT-ITM is normally used to generate one of three different kinds of topology, flat, N-level hierarchical and transit-stub [25]. A flat topology is non-hierarchical and is typically generated by placing vertices at random locations in a plane. An edge is added between a pair of vertices, based on a probability. Hierarchical models try to represent 'real-world' networks and reflect the notion that some nodes are more closely connected than others. A N-level hierarchical model generates a topology model recursively from a connected graph. Each node in a connected topology graph is recursively replaced with another connected graph. The transit-stub model is also a hierarchical model, but uses interconnected transit and stub domains. Transit domains represent a backbone infrastructure for a topology and have stub domains connected to them to represent the lower level sub-networks. The transit-stub model closely resembles the structure of the Internet. GT-ITM also provides five ways to generate edges in a topology graph. The different generation techniques are Pure Random, Waxman, Waxman 2, Exponential, and locality. Each of these types of generation has different characteristics and affects the end graph generated. An exact description of each type of topology generation algorithm is explained in greater depth in [25]. For network research purposes, the Pure Random distribution of nodes is typically used.

NS-2 is a complex simulator it attempts to model the whole network stack and provides results, which are a closer match to a 'real' network. NS-2 provides testing and demo scripts to verify its accuracy. However, the developers have stated, while efforts have been made to make sure the results are accurate, a developer is responsible for verifying their own results. Thus far, this is the only simulator seen to provide verification scripts to confirm the accuracy of the simulation results, in comparison to 'real' networks.

In terms of P2P protocol implementations, custom protocols are possible and Gnutella has been implemented in NS-2. A basic implementation of the BitTorrent protocol has been implemented in NS-2 [26]. However, some of the BitTorrent protocol has not been implemented, such as the HTTP protocol used for tracker communications has been omitted. Multi-tier topologies seem to be compatible with NS-2. NS-2 needs to be used with another software, to provide Java interoperability. The scalability of NS-2 is not known. NS-2 has an active community in terms of support via mailing lists and provides extensive documentation. However, NS-2 is composed of many packages and dependencies, which can make it difficult to install on some platforms. To fulfil some of our requirements NS-2 would have to be used with NAM. This would provide a simulator with visualisations, but without Java hooks for development.

2.2.2 PeerSim

PeerSim is a Peer-to-Peer simulator, which has been developed with scalability in mind. PeerSim was developed for the BISON project [27] and is used by the DELIS project [28]. PeerSim has cycle-based and event-based simulator engines. The cycle-based engine is simplified by ignoring the transport layer in the protocol stack and lacks support for concurrency. The lack of concurrency support means that the simulator sequentially gives control to each node in turn. The cycle-based engine allows the simulator to scale up to a larger number of nodes (one million nodes), but this does mean some accuracy is lost, in comparison with PeerSim's event-based engine. The accuracy of the cycle-based engine is acceptable for smaller simulations, however it is not known how much the error rate will increase with larger simulations.

PeerSim has some predefined protocols for P2P simulations, namely OverSat [29] [30], SG-1 [31] and T-Man [32]. Two-tier hierarchical models should be possible as the SG-1 protocol is based around a super-peer topology. PeerSim is written in Java, therefore protocols and simulator components are developed in Java. There are no visualisations of a simulation with PeerSim. PeerSim has some documentation and mailing lists, however the mailing lists are not used for support, but instead for release announcements and change logs. Documentation is provided in the form of Javadocs, tutorials and a protocol implementation guide. Publications using PeerSim and the three P2P protocols, namely SG-1, OverSat and T-Man are listed on PeerSim's website.

2.2.3 PlanetSim

PlanetSim is an overlay network simulation framework written in Java. It aims to provide easy transition from simulation code to prototype's by providing wrapper scripts in their simulator. PlanetSim does include a trivial P2P simulation sample simulation, however it is very basic. Chord [33] and Symphony [34] simulations exist and simulations are expected to scale to 100,000 nodes. PlanetSim supports static visualisations with GML or Pajek outputs of network topologies. PlanetSim is an event-based simulator, which uses a time-stepped method. This means there is a central clock that controls the events in a simulation.

Although Chord and Symphony P2P simulations exist in PlanetSim, it is not known if they are only search/discovery based simulations. Multi-tier simulations are available as third party components and thus should be feasible with this simulator. The documentation states that the Gnutella P2P protocol will be implemented. The accuracy of the simulations is not known and only two publications are listed on PlanetSim's website. PlanetSim does seem to have good variety of support, provided as Javadocs, tutorials, presentations, developer documentation and mailing lists.

2.2.4 OMNeT++

OMNeT++ is described as a simulation environment with GUI support. OMNeT++ is an event-based simulator. Like many simulators, it is component-based and modular in architecture. OMNeT++ also has an embeddable simulation kernel and utility class

written in C++, which is used for random number generation and statistics collection amongst other functions. OMNeT++ is mainly used for network simulations, but also for queuing networks, and multiprocessors. OMNeT++ itself does not provide components for network simulations; instead, there are simulation models and frameworks, which are used with OMNeT++. The INET framework [35] provides the simulation components for OMNeT++, the protocols include, IP, UDP, TCP, 802.11, Ethernet, PPP, IPv6, OSPF, and RIP amongst others. OMNeT++ also has a number of contributed models, which provide P2P protocols or simulations. The most interesting model for our needs is the P2P Swarming Protocol Simulation [36].

The P2P swarming add-on was developed as part of a final year project. It is based around the BitTorrent protocol, but only has a small subset of the functionality. The developers have stated that they are hoping to clean up the code and add further functionality in the coming months. A 1000 node swarming simulation was achieved with OMNeT++ and the P2P swarming add-on by the developer. Multi-tier topologies are supported by OMNeT++. Java interoperability is provided by the JSimpleModule [37], which is an extension that allows OMNeT++ modules to be written in Java. The JSimpleModule uses JNI-based wrappers for the corresponding C++ classes. Some coding conventions need to be adhered to for the Java used in a JSimpleModule. OMNeT++ does however provide an extensive GUI that provides visualisations of a simulation. There are several publications using OMNeT++ listed on their website. The accuracy of OMNeT++ in terms of a 'real-world' comparison is not known, but there is support in the form of a mailing list, a wiki, a forum and documentation available via their website.

2.2.5 OverSim

OverSim is an overlay network simulation framework for OMNeT++. OverSim shares many characteristics with OMNeT++, such as the GUI interface. OverSim can use the INET underlay model, which was derived from the INET framework used by OMNeT++. This provides OverSim with simulation models that can use the network layers from the MAC layer and above. Currently OverSim implements Chord, Kademila [38] and GIA [39] overlay network protocols.

Support for OverSim is provided as a mailing list and documentation for installation, usage and development. Simulations using 100,000 nodes with OverSim have been stated as feasible [40]. OverSim's underlay is exchangeable and thus provides flexibility. There are three different types of under-lays, simple, single host and INET. Only the INET underlay seems to be useful for our needs. OverSim only has one publication listed on their website, and the accuracy of the simulator is not known.

2.2.6 GPS

GPS is termed as a general purpose P2P simulator framework and is written in Java. GPS aims to provide accurate P2P simulations and allows for the implementation of custom protocols. GPS includes a download component to account for file transfers, which is a useful feature. It is described as a message-level simulator, which works at

the application-level of the network stack. GPS provides a GUI for the visualisations of a simulation.

The authors have implemented the BitTorrent protocol and state that "provides simulation models for BitTorrent, which, to the best of our knowledge has not been modeled functionally before." [14]. GPS appears to have the BitTorrent protocol implemented in full, including choking algorithms and support for file transfers. To maintain accuracy with 'real-world' performance GPS adds delay parameters for the TCP packets. The GPS simulator with the BitTorrent protocol has similar results to a 'real-world' test [14]. GPS uses topologies generated from GT-ITM using the transit-stub hierarchical topology to define a backbone for the network. Support is provided solely by the developer and Javadocs. GPS's website states that documentation is still being written. Only one paper is listed on the GPS website, and its popularity within the research community is not known.

2.2.7 AgentJ

AgentJ was developed as part of the Scalable Robust Self-organizing Sensor (SRSS) network project [41] for the U.S Naval Research Laboratory (NRL). AgentJ provides developers with a means of deploying Java applications into the simulator, with minor changes to the source code. It uses the popular and well-established NS-2 simulator. An advantage of this simulator is that code to execute the simulation is a normal Java application, which can be used in the 'real-world' with minor changes. AgentJ can also be used to run Peer-to-Peer Simplified (P2PS) [42] over NS-2. P2PS is a lightweight peer-to-peer infrastructure which can be used to develop peer-to-peer style applications, but without the complexity of similar architectures such as JXTA.

AgentJ lacks visualisation, thus, other software may be required to produce a visual representation of the simulation. The NS-2 simulator is a complex simulator and running AgentJ on top, adds more complexity. Support is provided by the developer and an extensive user manual. Although AgentJ may be a useful tool, the time taken to develop a full application to run with AgentJ, does not seem to merit many advantages over the implementation and testing on a suitable test bed. AgentJ would be better suited to being used while development is ongoing, or when a suitable test bed cannot be found.

2.2.8 P2PSim

P2PSim is an event-based simulator, which was developed as part of the IRIS project [43]. P2PSim's main goal was to make the understanding of P2P protocols easier. P2PSim is written in C++ and supports Chord, Accordion, Koorde, Kelips, Tapestry, and Kademlia P2P protocols.

The multi-tier capability of P2PSim is not known. P2PSim does not provide any simulation visualisations or a GUI. However, P2PSim can be used in conjunction with a third party GTK application to provide a GUI. Perl scripts are provided for the generation of graphs. The scalability of the simulator is in the order of 1000's of nodes. Support is provided in the form of an FAQ, a mailing list and a brief howto.

3. Installations and Tests

This section describes our experiences with the reviewed simulators outlined in Section 2. Each simulator was installed and executed, in an attempt to ascertain suitability for modelling the Portal-based P2P system.

3.1 NS-2 and NAM

NS-2 provides an all-in-one package, which has all the required dependencies to install and use NS-2, including NAM, TCL/TK and GT-ITM. The all-in-one package has an installation script, which will compile the dependencies and simulator. NS-2 installs and works without error, using the all-in-one package. To generate network topologies for NS-2, one must first create a topology using GT-ITM, which outputs a text file, using the Stanford Graphics Base (SGB) package. This topology output is converted to a format suitable for NS-2 using the provided tools. TCL scripts are then made which define the nodes, types of links and bandwidth. NAM can run the simulation with a visualisation of the topology and allows a user to modify the topology. The sample and tutorial simulations were setup and run with NS-2. Figure 2, shows the NAM GUI interface displaying a NS-2 simulation.

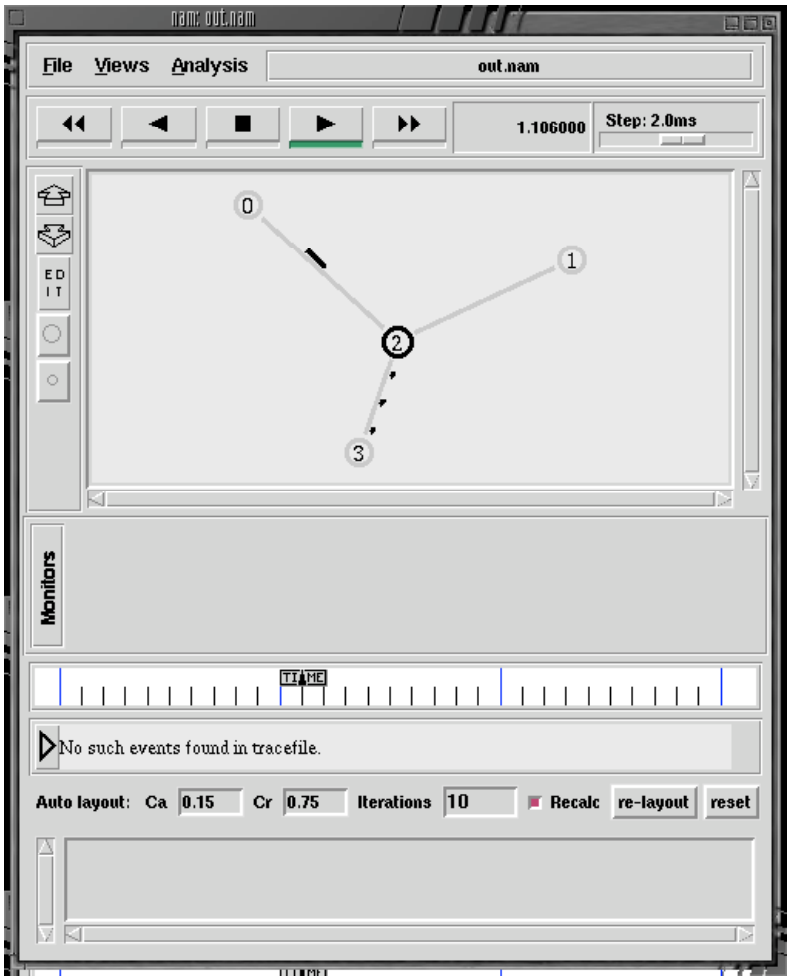


Figure 2, NS-2 simulation with NAM.

A standalone version of NS-2 is available from Gentoo Linux's [44] portage repository. The NS-2 package is not deemed stable by the Gentoo Linux community and thus is masked, which means it will not install by default. NAM is also available from Gentoo's portage repository, however NAM does not compile on the 64-bit version of Gentoo Linux. The standalone NAM package from the Gentoo repository can be compiled in a 32-bit `chroot` environment, however some library dependencies required resolving and instead it was easier to install the pre-compiled NAM binary in a 32-bit `chroot` environment.

3.2 PeerSim

PeerSim's installation was straightforward and without errors. PeerSim includes a number of sample simulations, which we tested. The Gossip-based aggregation protocol was used; whereby each node periodically selects a neighbouring node to communicate with, using an underlying overlay network.

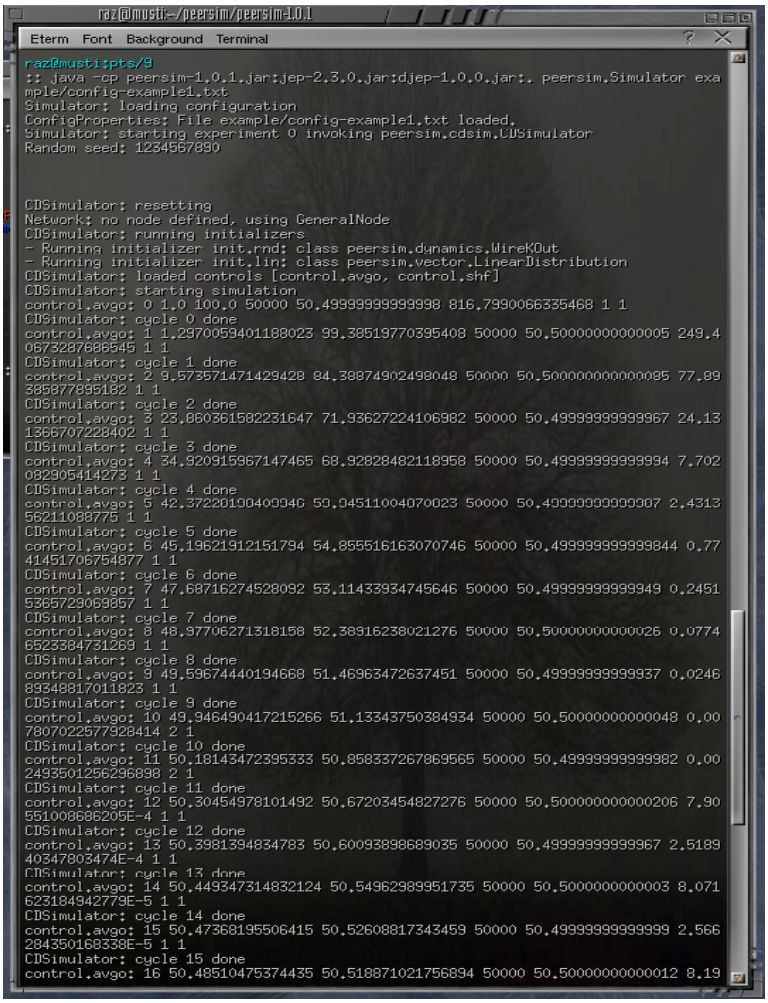


Figure 3, PeerSim running the sample P2P simulation.

Figure 3, shows the simple simulation carried out using the cycle-based engine. As a note most of the documentation focuses on the cycle-based engine and not the event-based simulator engine. Although the simulator works fine, the results output is not easy to understand and its usefulness for our needs is small.

3.3 PlanetSim

PlanetSim is packaged as a Java application. Installation was straight forward and without error. PlanetSim divides a simulation into two main sections, the overlay network and associated services. The overlay networks for the simulation are algorithms such as Chord and Symphony, while services, such as Distributed Hash Tables (DHT), are run on top of a configured overlay network. PlanetSim's distinction within a simulation between an overlay network and services, allows a developer to test and experiment with different overlay networks and different services.

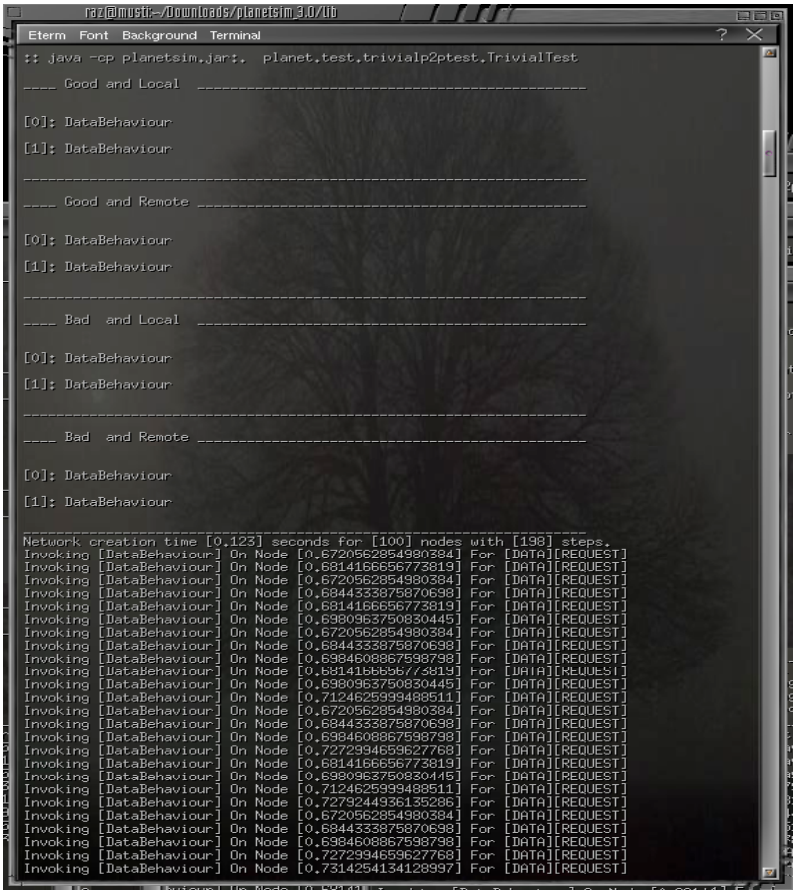


Figure 4, PlanetSim running the trivial P2P simulation.

The included trivial P2P sample simulation, which was run with 100 nodes, is shown in figure 4. Although PlanetSim provides static visualisations with GML or Pajek outputs, it does not provide any mechanisms for gathering statistics, resulting in developers having to implement their own methods for collecting information about a simulation.

3.4 OMNeT++

OMNet++ and its associated components installed without error. The GUI interface runs a visualisation of the network and seems to work well with 1000 nodes being rendered in the GUI. There are no P2P examples or settings, bundled with the default OMNET++ installation. The swarming extension seems to be in the very early stages

of development and no documentation, or installation instructions were provided. The swarming extension developer does have a blog, which lists some of the bugs, and issues that have been found in the swarming module.

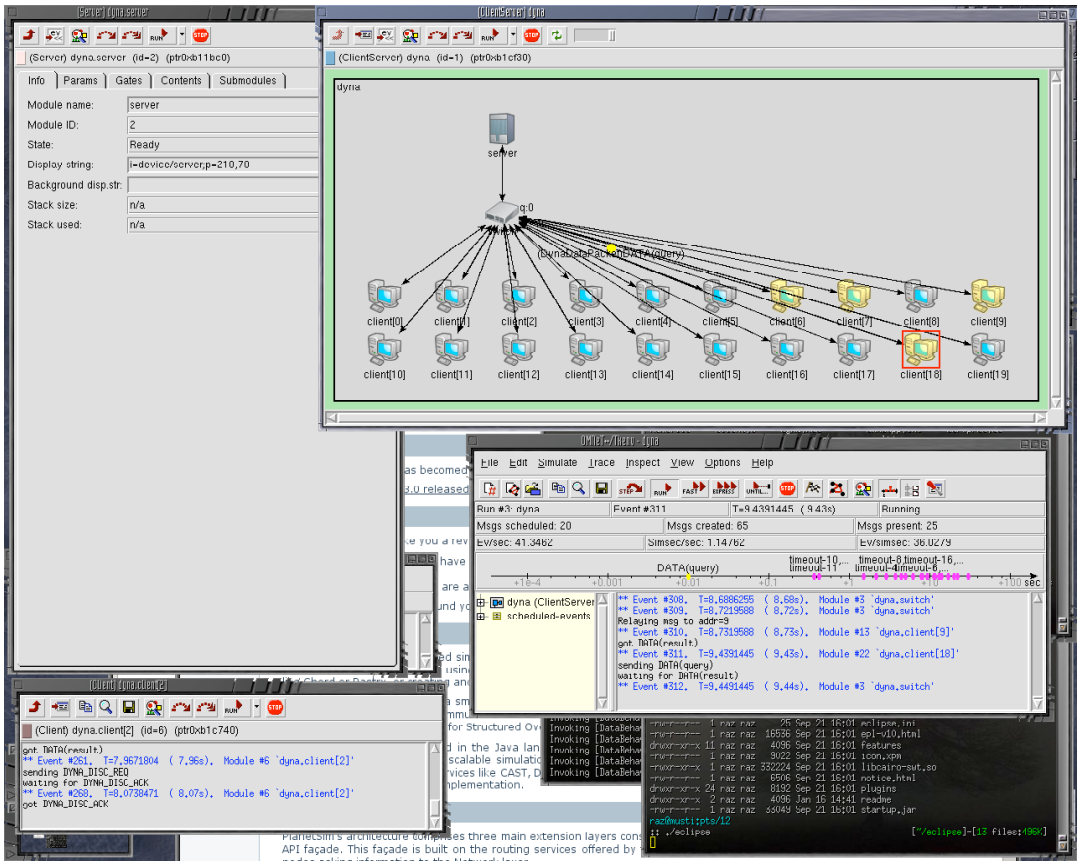


Figure 5, OMNeT++ running a client/server simulation.

An attempt was made to make a simulation using the JSimpleModule, however during the initial stages of development it became obvious that creating the module was a harder task than expected. There would have to be a mix of C++ and Java to develop a simulation. It was hoped that the Java modules made could be used elsewhere or in the actual implementation of the Portal-based P2P system, but the restrictions placed on the coding conventions that are allowed in the JSimpleModule mean that this is not possible. The coding conventions needed for the JSimpleModule include:

- Explicit memory management – Java wrapper objects do not delete the underlying C++ object,
- No Java object identity – the "==" operator cannot be used to compare two objects. Instead, the underlying C++ pointers need to be compared,
- No polymorphic return types – When an object is returned the C++ object is wrapped in a Java object of the return type.

A simple simulation with 20 nodes and a single FTP server producing traffic was run, which can be seen in Figure 5.

3.5 OverSim

OverSim's installation was lengthier than OMNET++, because the required dependencies, namely OMNeT++ and the INET Framework, needed to be installed first. OverSim recommends that GCC version 4.x be used to compile the simulator. An installation with GCC version 3.3.6 was attempted, as this was the stable and default version of GCC available on the development machine. Compilation with GCC version 3.3.6 failed, however OverSim was compiled with GCC version 4.1.x successfully. Although compilation completed without errors the simulator failed to execute and produced the output below.

```
 [~/sim/OverSim-20061215/Simulations]:: ../bin/OverSim
OMNeT++/OMNEST Discrete Event Simulation  (C) 1992-2005 Andras Varga
Release: 3.3, edition: Academic Public License.
See the license for distribution terms and warranty disclaimer
<!> Error during startup: Cannot load library
   '/home/raz/sim/OverSim-
20061215/Simulations/./../Overlay/Chord/Chord.so':
/home/raz/sim/OverSim-
20061215/Simulations/./../Overlay/Chord/Chord.so: undefined symbol:
 _ZNK18BaseOverlayMessage7getTypeEv
End run of OMNeT++
```

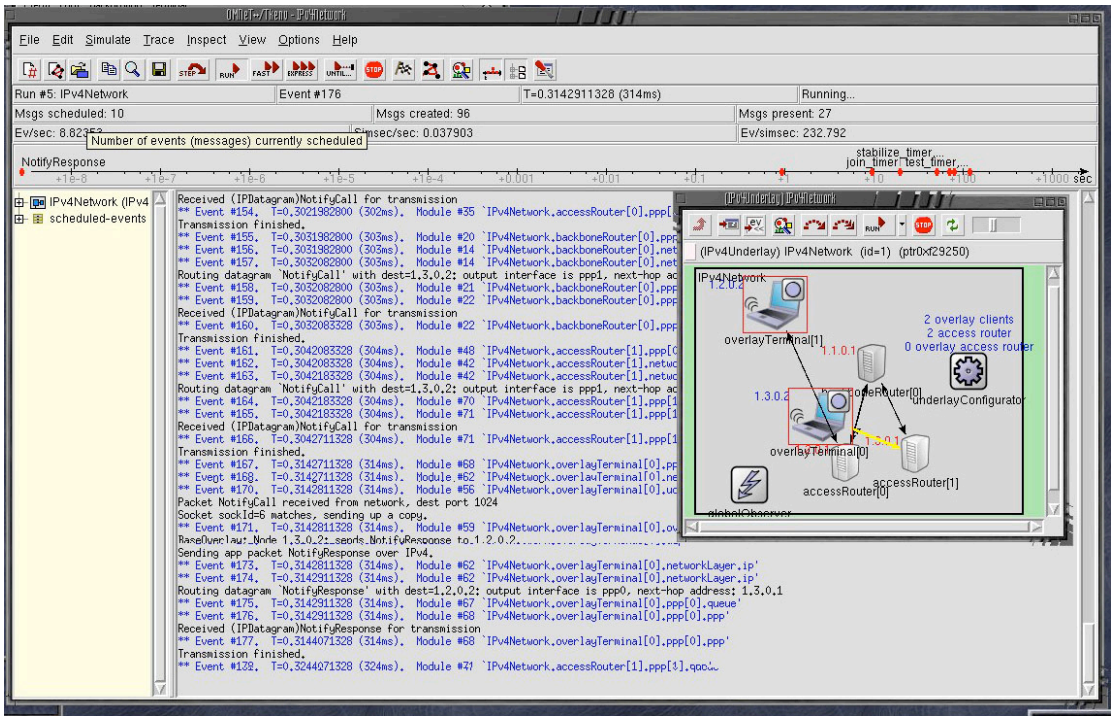


Figure 6, OverSim running Chord simulation.

The error occurred when trying to load a library for the Chord overlay network. An email was sent to the OverSim mailing list detailing the issue. A patch was provided by the OverSim mailing list, which has been applied to the installed version. The developers stated that there might be some issues with SE-Linux, which may not be allowing libraries to be loaded at runtime. The sample Chord P2P simulations where setup and used with OverSim, which can be seen in Figure 6. OverSim has the same issues with the JSimpleModule as OMNeT++.

3.6 GPS

The installation and execution of GPS requires Apache Ant. GPS provides both CLI and GUI user interfaces, however the CLI interface has no documentation. Stability issues were found with GPS. The simulator hangs when running a simulation with over three hundred nodes. Although the total number of nodes in a simulation can be defined, the number of active nodes transferring files is configured separately. There is a limitation when there is more than hundred active nodes transferring files. Some parts of the GUI are also not stable and the GUI seems to crash/lock, for example, the stop button does not always work. The BitTorrent protocol within GPS is not an implementation of the latest version of the BitTorrent protocol, which does not use DHTs and UDP to transfer files. GPS includes the option to define the number of BitTorrent trackers in the network. GPS has an event configuration file, which defines the order and timing of events in the simulation. There is also a documents configuration file, which has the size and priority of a document for each node in the simulation. These files are generated from undocumented classes in the simulator.

We were unable to generate topologies using GT-ITM, which matched those packaged with GPS. The sample topologies bundled with the simulator seem to be missing stub nodes (i.e. those nodes, which are situated on the edge of the topological graph). Instead, GPS's topologies stop at the stub-domains attached to the transit-domains. We contacted the developer asking how the provided topologies have been produced. We are still waiting for a reply from the developer. It was stated in [14] that the simulator developer, removed the nodes at the edge of the topologies, but reasons for this modification were not given.

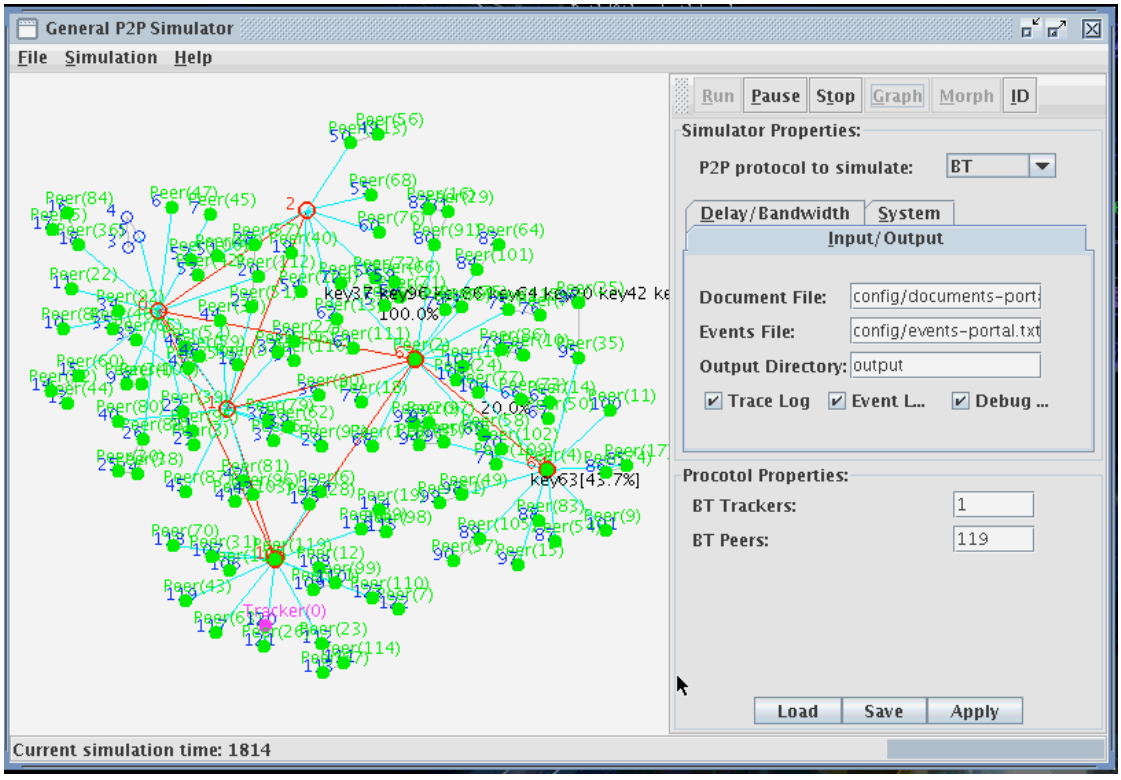


Figure 7, GPS running P2P simulation.

A GT-ITM topology was generated with 120 nodes and loaded into GPS successfully, even though our graph had additional nodes at the edge of the topology graph, shown in figure 7, which differs from the topologies included with GPS. An event configuration file was generated that invoked an event at a particular node every ten seconds. In addition, a document configuration file was generated which defined a 500 Mbyte file, which was in used in the simulation as data. This file also defined which nodes have a copy and their priority in the network for serving the file. The simulation was configured with a single tracker and 119 active peer nodes. Unfortunately, the simulator was not able to complete the test. An exception was shown relating to the number of entities in an array, which tracks the chunks of a file for each peer in the network. Lowering the number of active peers to 64 and defining a 250 Mbytes document, allowed the simulation to run for a longer period, before halting. The simulation with 64 active peers did not produce any errors or results. It has been noted that the provided sample simulation on a graph of 512 nodes still only has 64 active nodes. This leads us to believe that there is a scalability issue with this simulator, when there are 100's of nodes.

It has also become apparent that the GPS simulator is tightly coupled to the BitTorrent protocol, making it difficult to implement our own protocol. This issue has also been seen in [45] when trying to implement the Chord protocol.

3.7 AgentJ

AgentJ's documentation is extensive and its installation is not straightforward. A NS-2 extender package was found which adds various extra features to NS-2. One of the additions that the NS-2 extender packaged offered was AgentJ. The extender package has an automated script that installs the dependencies and modifies the original NS-2 automatically. The manual method of installation following the user manual is very lengthy and requires a user to manually patch source code and edit `make` files.

When modifying an application to run on AgentJ the Java socket package needs to be changed. AgentJ overrides the default Java socket, so that data can flow through the simulator. In addition, NS-2 needs to be configured with the bandwidth on each link, network topology and other network characteristics using TCL scripts. AgentJ's user manual describes this process [46]. It was noted in AgentJ's documentation that an underlying library for AgentJ, called Protolib, does not support TCP in the simulation environment [46]. Although the documentation does state that TCP support is planned.

3.8 P2PSim

P2PSim requires GCC version 2.95.3, which is very old. As a result finding a suitable machine to compile P2PSim on was challenging. P2PSim did not compile with the latest version of GCC (version 4.x). An old version of the Debian Linux distribution named 'Woody' found to have GCC version 2.95.4 although the versions did not match P2PSim was compiled successfully.

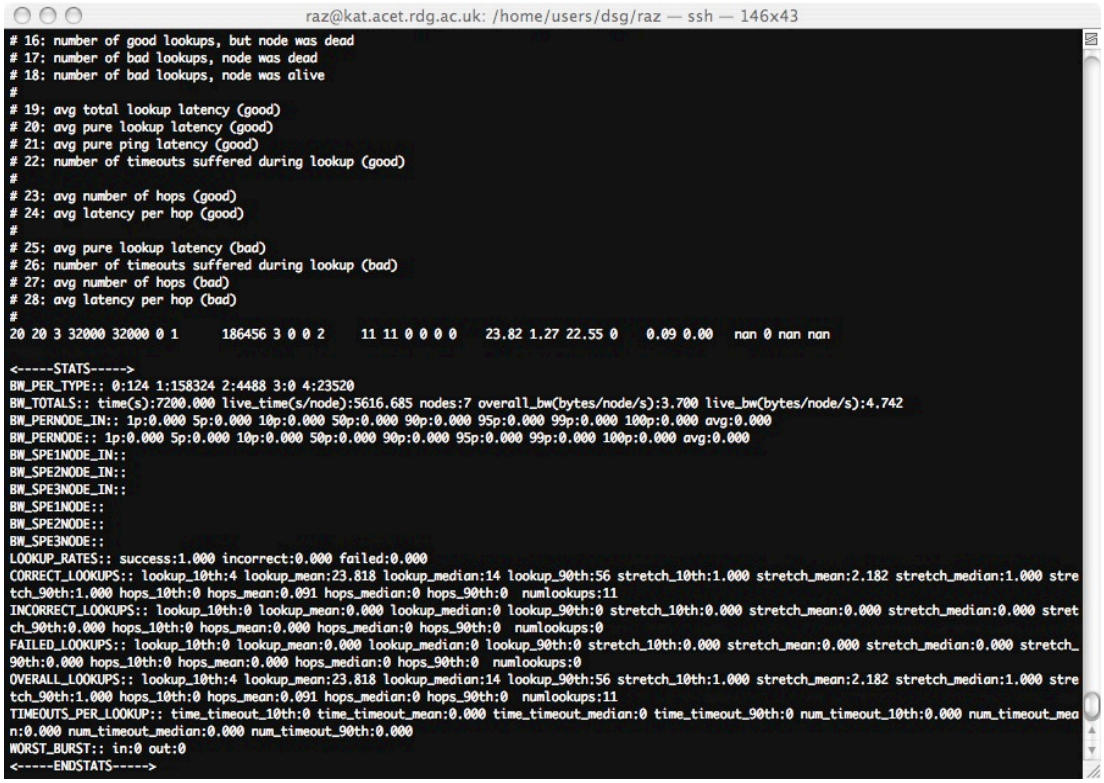


Figure 8, P2PSim running a Kademila P2P simulation.

P2PSim has some dependencies that are no longer found in Debian's repository because of their age. Thus, a combination of manual compilation and packages was used to deal with the dependencies. The gmp-4.1.2 package [47], which is used for floating point calculations, was compiled by hand. A final dependency of the Standford Graphic Base (SGB), was not installed for the optional graph creation capabilities. Debian packages were found for SGB, but did not install due to the old age of the distribution, which is no longer supported. The example simulation using Kademila DHT with 7 nodes was run, and can be seen in Figure 8. The issues experienced with P2PSim are most likely due to the age of P2PSim. Its last update according to the project was in 1995 [21].

4. Summary and Conclusions

4.1 Summary

The Portal-based P2P system is designed to socially manage large data sets, such as those produced by the SDSS project. Network simulators provide a virtual network environment, which can aid the development of new P2P systems, by overcoming impracticalities of developing such a system on a real network. The aim of this investigation was to discover if a suitable simulator could be used to help test and design our P2P system.

The focus of this report has been to assess simulators suitable for the needs of the Portal-based P2P project. The report presents a brief overview of our Portal-based

P2P system and our criteria. Several simulators were reviewed, installed and tested. The report highlighted issues found when testing the simulators.

A set of criteria was made that outlined our needs from a simulator. The core criteria was to have support for custom protocols, facilitate multi-tier topologies, provide visualisations, output accurate results, have good support and preferably interface with Java for development. Other criteria used for the simulators included:

- The scalability of the simulators,
- Distributed simulations to decrease the time taken for a simulation,
- The statistical output from a simulator,
- P2P swarming protocol/capability.

NS-2 is a well-established network simulator within the research community, but is complex and can be difficult to install and use. Visualisations for NS-2 are provided by another application called NAM. NS-2 has a large set of dependencies, including TCL/TK and GT-ITM, but is well documented and supported. GT-ITM is a standard tool used to generate different types of network topologies. GT-ITM is used by NS-2 and other simulators such as GPS. NS-2 also forms the base for AgentJ.

PeerSim is a Java based simulator specifically designed for P2P systems. PeerSim has three P2P models included, namely OverStat, SG-1, and T-Man. PeerSim does not provide any visualisations and support is provided in the form of Javadocs and online manuals. PeerSim has two types of simulation engines, one is cycle-based the other event-based. The cycle-based simulations can scale further with the number of nodes, but does not provide accurate results with larger models. The event-based engine provides results that are more accurate, but it is not as well documented.

PlanetSim is another Java based simulator, which provides wrapper scripts for Java applications so that they can be tested in a simulation environment. Chord and Symphony P2P simulations exist and PlanetSim can scale to 100,000 nodes. Static visualisations are provided with GML or Pajek outputs, however no visualisations of the simulation are provided at runtime. PlanetSim does not provide any default mechanisms to gather results from a simulation, thus a developer needs to implement their own methods to gather statistical information.

OMNeT++ is a simulation environment with GUI support. It is an event-based simulator and uses the INET framework to provide simulation components and protocols. A P2P swarming protocol extension for OMNET++ is in the early stages of development. Java interoperability is provided with the use of the JSimpleModule, however JSimpleModule has certain conventions, which need to be implemented in the code. The JSimpleModule cannot be used to create a entire simulation therefore a mixed development environment of C++ and Java is required. OMNeT++ provides good support in the form of an active mailing list, a wiki, a forum and documentation available via their website.

OverSim is an overlay network simulator based on OMNeT++. OverSim shares many characteristics with OMNeT++. Support is provided by a mailing list and through documentation for installation and development. OverSim uses an INET underlay, which is derived from the INET framework used for OMNeT++. OverSim had errors

during installation, which required a patch from the developers. The JSimpleModule could be used with OverSim, but has the same limitations and characteristics when used with OMNeT++. Currently OverSim implements the Chord, GIA, and Kademila network overlay protocols.

GPS is termed as a general purpose P2P simulator framework and is written in Java. GPS aims to provide accurate P2P simulations and allows for the implementation of custom protocols. GPS includes a download component to account for file transfers. It is described as a message-level simulator, which works at the application-level of the network stack. GPS provides a GUI for the visualisations of a simulation. GPS had some stability issues during testing and lacks documentation. The implemented BitTorrent protocol is tightly coupled to the simulator making it difficult to implement a custom protocol.

AgentJ provides developers with a means of deploying Java applications into the simulator, with minor changes to the source code. It uses the popular and well-established NS-2 simulator. AgentJ executes normal Java applications within the simulation and only requires overriding of the Java socket package. AgentJ adds more software layers and libraries to an already complex simulator, NS-2. AgentJ provides documentation in the form of a user manual and support is provided by the developer. The documentation states that TCP is not supported in the simulation environment, but is a planned addition.

P2PSim is an event-based simulator, and its main goal was to make the understanding of P2P protocols easier. P2PSim is written in C++ and supports Chord, Accordion, Koorde, Kelips, Tapestry, and Kademlia P2P protocols. P2PSim's dependencies are dated; making the installation of the simulator a difficult task. The P2PSim project is currently static, with its last update in April 2005. P2PSim does not provide any visualisations but does provide a means to generate graphs from the results. Support is given in the form of a mailing list, FAQ and a howto.

4.2 Conclusion

Although network simulators can be used for P2P systems, their usefulness for our Portal-based P2P system is not clear. All of the simulators, except AgentJ and PlanetSim, require a developer to implement a P2P system specifically for a particular simulator. Thereby increasing the development time for application developers. However, simulators such as AgentJ and PlanetSim can be used once a suitable stage has been reached in the development cycle of the P2P system. Practical issues and limitations arise, as AgentJ is a complex simulation environment and the documentation states TCP is currently not supported. PlanetSim provides a very similar environment, but does not provide any default means to gather results. None of the simulators completely fulfils our requirements, however AgentJ and PlanetSim are the best candidates.

As with any simulation the results need to be verified. In our case, the simulation would need to be compared with the Portal-based P2P system in an emulated environment, typically on a controlled and stable 'real' network. This would require a

simulation implementation and a Portal-based P2P application implementation for a comparison, which increases the development time and effort required.

In light of this report, which reviews and tests several simulators, it is concluded that the effort and time taken to accurately implement our Portal-based P2P system in a simulator would be better spent implementing the 'real' system. It is preferred to use simulators such as AgentJ or PlanetSim, during the development cycle, as they do not require a specific simulation implementation. This ensures that the code developed does not need to be forked for a particular simulator, reducing the probability of errors and increasing the maintainability of the project. The wrapper style simulators are useful for testing our system with scenarios that may not be practical or possible on an emulated test bed.

References

[1] Schollmeier, R, A definition of peer-to-peer networking for the classification ofpeer-to-peer architectures and applications, IEEE, P2P'01, p 101-102, (2001).
[2] Ripeanu. M., Peer-to-Peer Architecture Case Study: Gnutella Network, IEEE P2P'01, p. 0099, (2001).
[3] BitTorrent, http://www.bittorrent.com
[4] Kazaa, http://www.kazaa.com/us/index.htm
[5] Heckmann. O., Bock. A., The eDonkey 2000 Protocol, KOM Technical Report, (2002).
[6] LimeWire, http://www.limewire.com/english/content/home.shtml.
[7] Freenet, http://freenetproject.org/.
[8] BBC News. BitTorrent battles over bandwidth, http://news.bbc.co.uk/1/hi/programmes/click_online/4905660.stm.
[9] The Register. The BitTorrent P2P file-sharing system, Detailed measurement study, http://www.theregister.co.uk/2004/12/18/bittorrent_measurements_analysis/.
[10] Sloan Digital Sky Survey, http://www.sdss.org.
[11] SDSS Storage Server Technical Note, http://home.fnal.gov/~yocum/storageServerTechnicalNote.html.
[12] Baker. M.A., and Grove. M., Tycho: A Wide-area Messaging Framework with an Integrated Virtual Registry, Special issue of the Journal of Supercomputing, G.A. Gravvanis, J.P. Morrison and G.C. Fox, Springer US. (2006), ISSN 1573-0484.
[13] NS-2, http://www.isi.edu/nsnam/ns/.
[14] Yang. W., and Abu-Ghazaleh. N., GPS: a general peer-to-peer simulator and its use for modeling BitTorrent. Proc MASCOTS'05. IEEE (2005), 425-432.
[15] OMNeT++, http://www.omnetpp.org/.
[16] OverSim: The Overlay Simulation Framework. http://www.oversim.org/.
[17] NAM: Network Animator, http://www.isi.edu/nsnam/nam/.
[18] PeerSim, http://peersim.sourceforge.net/.
[19] PlanetSim, http://planet.urv.es/planetsim/.
[20] AgentJ, http://cs.itd.nrl.navy.mil/work/agentj/index.php.
[21] P2PSim, http://pdos.csail.mit.edu/p2psim/.
[22] REAL Network Simulator, http://www.cs.cornell.edu/home/skeshav/real/overview.html.
[23] VINT Project, http://www.isi.edu/nsnam/vint/index.html.
[24] GT-ITM, http://www.cc.gatech.edu/projects/gtitm/.
[25] Ken Calvert, Matt Doar and Ellen W. Zegura. Modeling Internet Topology. IEEE Communications Magazine, June 1997.
[26] Simulation of BitTorrent Peer-ot-Peer (P2P) Networks in NS-2, http://www.tu-harburg.de/et6/research/bittorrentsim/index.html.
[27] BISON, http://www.cs.unibo.it/bison/.
[28] DELIS, http://delis.upb.de/.
[29] Montresor. A., Jelasity. M., Babaoglu. O., Robust Aggregation Protocols for Large-Scale Overlay Networks, IEEE, DSN'04, p 19-24, (2004).
[30] Jelasity. M., Montresor. A., Epidemic-Style Proactive Aggregation in Large Overlay Networks, IEEE, ICDCS'04, p 102-109, (2004).

[31] Montresor. A., A Robust Protocol for Building Superpeer Overlay Topologies, IEEE, In Proceedings of the 4th International Conference on Peer-to-Peer Computing, (2004).
[32] Jelasity. M., Babaoglu. O., T-Man: Fast gossip-based construction of large-scale overlay topologies. Technical Report UBLCS-2004-7, University of Bologna, Department of Computer Science, Bologna, Italy, May 2004.
[33] The Chord Project, http://pdos.csail.mit.edu/chord/.
[34] Manku. G., Bawa. M., and Raghavan. P., Symphony: Distributed Hashing in a Small World, Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems, pp. 127–140, (2003).
[35] INET Framework, http://www.omnetpp.org/staticpages/index.php?page=20041019113420757
[36] P2P Swarming Protocol Simualtion, http://me55enger.net/index.cgi/software/swarm
[37] OMNeT++, JSimpleModule, http://www.omnetpp.org/pmwiki/index.php?n=Main.JSimpleModule
[38] Maymounkov. P., and Mazires. D., Kademlia: A peer-to-peer information system based on the xor metric, IPTPS 2002. Revised Papers, vol. Volume 2429/2002, pp. 53–65, (2002).
[39] Chawathe. Y., Ratnasamy. S., Breslau. L., Lanham. N., and Shenker. S., Making gnutella-like P2P systems scalable, ACM, SIGCOMM ’03, pp. 407–418, (2003).
[40] Baumgart. I., Heep. B., Krause. S., OverSim: A Flexible Overlay Network Simulation Framework, IEEE, GI'07 in conjunction with IEEE INFOCOM'07, (2007).
[41] Scalable Robust Self-organizing Sensor (SRSS) network project, http://cs.itd.nrl.navy.mil/work/srss/index.php
[42] P2PS, http://www.trianacode.org/p2ps/.
[43] P2PSim, http://project-iris.net/.
[44] Gentoo Linux, http://www.gentoo.org/.
[45] Naicken. S., Basu. A., Livingston. B., and Rodhetbhai. S., A Survey of Peer-to-Peer Network Simulators, Proc The Seventh Annual Postgraduate Symposium, Liverpool, UK, (2006).
[46] AgentJ Documentation, http://pf.itd.nrl.navy.mil/agentj/agentj.pdf.
[47] GMP, http://gmplib.org.