The Two-Step Overlay Network Simulation Approach

A Framework for Message- and Packet-Level Simulation

Hannes Birck, Oliver Heckmann, Andreas Mauthe and Ralf Steinmetz

KOM Multimedia Communications Lab
Department for Electrical Engineering and Information Technology & Department for Computer Science
Darmstadt University of Technology
Merckstr. 25, 64283 Darmstadt, Germany

{birck,heckmann,mauthe,steinmetz}@kom.tu-darmstadt.de

Abstract— **In this paper a tool collection is introduced that can be used to analyse the effect & requirements of P2P applications on application and on network layer. P2P applications are complex and deployed on a large scale, pure packet level simulations do not scale well enough to analyse P2P applications in a large network with thousands of peers. It is also difficult to assess the effect of application level behavior on the communication system. We therefore propose an approach starting with a more abstract and therefore scalable application level simulation. For the application layer a specific simulation framework was developed. The results of the application layer simulations plus some estimated background traffic are fed into a packet layer simulator like NS2 (or our lab testbed) in a second step to perform some detailed packet layer analysis such as loss and delay measurements. This can be done for a subnetwork of the original network to avoid scalability problems.**[1]

1. INTRODUCTION

SIMULATIONS are currently a popular way to investigate crucial issues and possible system behavior within communication networks. However, a good design for a simulator that reflects the network behavior realistically is not so easy. In this paper a novel approach for a application layer simulation (ALS) is introduced. This is an extension of the packet level simulation toolset KOM ScenGen [1]. This new approach presents a way to analyze behavior at application layer, as well as considering the underlying communication system. The introduced framework is applicable for all systems based on application layer overlays.

A further important item is the fast progress in the area of overlay applications - particularly peer-to-peer. Many new protocols and methods are developed, but there is hardly a possibility to make a comparison between them. Scalable simulation can help comparing and evaluating these protocols and applications.

The simulation model uses two steps, the first step is simulating the behavior at application layer. The result is then considered in the second step, viz. the packet level simulation. By these two steps approach we have the possibility to avoid problems related to performance, and the the accuracy and the detail level of the modeling as for instance discussed in [2].Thus the system complexity is kept low while still maintaining a realistic model at each of the corresponding levels. This is in contrast to many of the current peer-to-peer simulator designs that mostly concentrate on the functionality of a peer-to-peer system and do not explicitly consider a realistic network environment. Hence, they stress protocol behavior rather than looking at the influence of network parameters such as delay, number of routers and links, geographical locations, distances, etc. A good overview for peer-to-peer demonstrators and simulators can be found at the P2PJournal web page [3].

[1]This work was supported by E-Next project, a consortium of 41 european institutions to build a sucessfull European Research Area (FP6/2002/IST/1).

With this ALS approach we aim at an exact mapping of the protocol *and* additionally a realistic network environment without neglecting packet level details crucial for evaluating the influence of application behavior on the underlying network. Additional it is possible to compare different protocols, taking account realistic network conditions.

The remainder of this paper is organized as follows. Section 2 describes the packet-level simulation and emulation toolset. Section 3 presents the novel application layer simulation approach, and Section 4 finished with the summary and the conclusion.

2. SYSTEM OVERVIEW

In this section we give an overview how networking experiments are supported by our tool collection. We start with some terminology and discuss how traffic can be represented in experimentation environment. Next, the different steps of conducting an experiment are described from beginning to end. In the next section, we will focus on the application level simulation step which is the main contribution of this paper.

2.1 Terminology and Traffic Description

The term **traffic** is used to describe the amount of bits that are transmitted over one link or are sent by a node. With the term traffic we always mean Internet (IP) traffic. Traffic can be modeled at different layers with different degrees of abstraction.

On the lowest layer IP traffic can be modeled as a **series of packets**. Each packet is characterised by a generation time and size plus source and target node and port plus protocol number. Traffic can also be modeled on higher more abstract layers. If traffic is aggregated in time we call this the **intensity layer** which specifies traffic as the number of bytes transmitted between a source and destination(s) or on one link in a single period of specified length. The information about the individual packet sizes is lost this way. It is non-trivial to split an intensity into individual packets again. Traffic matrices are an example that typically use traffic intensities. Also some trace files specify traffic intensities and some self-similar traffic models specify how to generate traffic intensities.

If traffic is not aggregated in time but instead by context we speak of the **flow layer**. Each flow generates a series of packets with a flow-type specific algorithm. A CBR flow transmits packets of fixed size in constant intervals. A greedy TCP Reno flow transmits packets as fast as possible using the TCP Reno flow and congestion control algorithm. The advantage of flow layer traffic is that it is obviously very powerful and memory efficient since all packets belonging to a flow can be described by a few flow parameters. However each flow type (CBR, greedy TCP, etc.) has a very different set of parameters and the flow algorithm has to be implemented both in the simulator and traffic emulator. All flows have a start time and a node/port pair. The greedy TCP source has the following additional parameters: *Packet size, Amount of data to be transferred and TCP algorithm parameters.*

The next highest layer is the **session layer**. A session consists of a number of closely related flows or intensities. A simple IP telephony session for example might contain a number of CBR flows following each other with switching directions. A session can be seen as the runtime instance of one application.

The highest layer - the **application mix layer** - models how many sessions of which traffic model respective application are generated in one edge node (e.g. 40 IP Telephony, 20 Peer-to-Peer and 100 WWW sessions). The application mix is specified in the node & link property step (see below).

Our application level simulation framework (see Section 3) breaks the application mix layer information down to session and flow information.

In **network simulation** computer models of real network components are used to estimate the behavior of the network to some input considering to typical networking parameters such as loss, delay, throughput. Network simulators like NS2 [4], JavaSim [5], OpNet [6] etc. are uses for network simulation. Our presented approach currently uses NS2 for packet level simulations and our own framework for the application level simulations (see Section 3). Contrary to simulations, in a **real-world** or a **testbed experiment** the behavior of a network to specific input is observed based on measurements made in a real physically existing computer network, either a testbed, research network or production network.

2.2 Conducting an Experiment

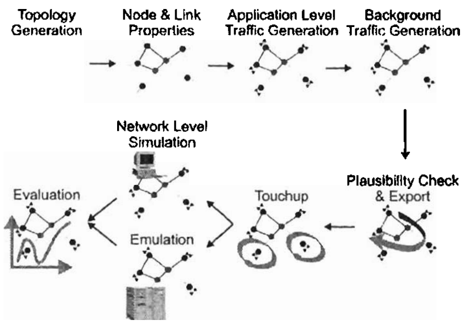
Topology Generation
Node & Link Properties
Application Level Traffic Generation
Background Traffic Generation
Network Level Simulation
Plausibility Check & Export
Evaluation
Touchup
Emulation

Figure 1 – Conducting a Network Experiment

Figure 1 shows the different steps of conducting a network experiment. First a **topology** is **created** either manually or automatically. To support this, we offer a library of real-world topologies [7] and a converter for different topology generators like TIERS [8], BRITE [9], [10], GT-ITM [11] and Inet [12]. Topologies can also be created with a special GUI. See also Section 3.1.

It has been also investigated how to choose the parameters of the topology generators in order to obtain realistic topologies. The results show that the topology generators above can indeed produce realistic topologies with respect to outdegree distribution, the hop-plot and some other metrics.

Next, the **properties** of the **links** and **nodes** are set manually or automatically. These properties include

- Delay of a link
- Bandwidth of a link
- Queueing algorithm and queue size of a link
- Traffic properties of a node

These properties can be set automatically with a script or manually using the GUI mentioned above.

In the next step we run the **application level simulation** that is described in detail in Section 3. It uses the topology and traffic information to simulate the application behaviour on that topology. Currently this step focuses on simulating the behaviour of P2P applications but other applications could be supported as well. This step generates flows and sessions that represent realistic P2P traffic. For some experiments this might be everything the researcher is interested in, in that case the experiment can stop after this step. Otherwise **background traffic** is added in the next step to run network level experiments (simulation or testbed experiment) later on. For adding background traffic we implemented some smaller traffic models, for example an aggregated WWW model based on the traffic generator of Kramer [13].

After the traffic generation steps are completed, the resulting experiment setup is exported. During the export a **plausibility check** can be run which checks parameters critical for the experiment for plausibility. An example would be estimating the bandwidth necessary for the generated traffic and comparing it with the available bandwidth. We implemented two algorithms to estimate the used bandwidth, one uses fixed rates for the TCP connections and given[2] loss probabilites while the other one is more sophisticated and based on M/M/1 queues and the TCP formula [14]. If much more bandwidth is needed than offered the operator might want to change the scenario parameters before investing time in the actual simulation or testbed experiment. After the plausibility check the scenario is exported to NS2 and/or the testbed:

The **NS2 export module** can automatically create an OTcl file for NS2 called *run.tcl* that sets up the topology and the traffic sources and starts them. To allow the user to finetune the setup process for his needs we do not directly configure NS2 in the *run.tcl* script but instead call setup functions that are defined in a second OTcl file *header.tcl*. Usually, the operator only has to adapt the *header.tcl* to his specific scenarios needs while the *run.tcl* file can be generated automatically and does not have to be changed.

The **testbed export module** is written for the testbed of our lab that consists of 24 FreeBSD routers. It can be easily adapted to similar testbeds. The export module of the scenario generator creates a number of configuration files and scripts. When the masterscript is started it sets up the testbed completely automatic. When a second script is started the experiment is also started automatically.

First SSH host keys on the machines are exchanged. Next the DNS and DHCP server on the control machine are configured and restarted, then all machines in the testbed are rebooted. The IP addresses of their interfaces are distributed by the DHCP server, the DNS server allows us to address the machines with the same names as in the scenario file. Next the switch is configured automatically, VLANs are set up to represent the links of the topology. Unused network interfaces are put into dummy VLANS. Because VLAN headers will be added to every packet we had to modify the Ethernet network drivers because otherwise full-size ethernet packets could not be sent. We use a shortest path algorithm to calculate the routes and set up static routing in all nodes. After that ALTQ [15] and dummynet [15] configuration files are distributed to all nodes and ALTQ is started. ALTQ is a traffic management software that enables certain QoS mechanisms on PC-based routers. Dummynet can be used to emulate a wide variety of network conditions by applying bandwidth and queue size limitations and emulate delays and losses. Then the configuration files for the traffic emulator tool written introduced in [16] are distributed to all nodes and can be started automatically. The

[2]Estimated by the experimenter

clocks of our testbed machines are synchronized by a GPS receiver.

After the export step the packet level simulation or testbed experiment can be started and evaluated.

3. THE APPLICATION LAYER SIMULATION

In this section the Application Layer Simulation (ALS) step is described in more detail. In this step application messages instead of IP packets are analyzed. Every message has a well-defined size and content. In the context of the simulation, the underlying network structure is based on realistic physical structures respectively on Internet structures. Since this approach concentrates on a higher abstraction level it is possible to avoid the problems that arise in the simulation of large networks [17], [18], [2]. With the exact traffic model for the application layer, the ALS system is used for packet level traffic generation. Additional ALS can accomplish studies for analysis and optimization at application layer. It is beneficial to do this kind of studies with realistic simulation environment.

The ALS framework itself is implemented in C++ and is based on the ComNets Class Library (CNCL) [19]. CNCL is an object oriented library for event driven simulations. For graphical task the Boost Graph Library (BGL) [20] is used.

The application level simulation framework is roughly subdivided into four parts, the physical topology creation, the user/data model, the traffic forwarding and the protocol. In the following each part will be discussed in more detail.

3.1 Physical Topology Creation

We start with the description of the underlying network topology for the ALS framework. That means a real-world network topology and not the overlay topology constructed by the application protocol as discussed in the Section 3.4. The topology is usually significantly influencing the outcome of the simulation. Important properties such as end-to-end delay and packet loss depend on the used network topology. This topic is discussed in more detail in the Traffic Forwarding Section 3.2 paragraph.

In order to proof the functionality of a particular peer-to-peer protocol in general, it is sufficient to use a small topology, which is optimized for the considered problem. There are a number of demonstrators for special peer-to-peer protocols [3]. For an effective analysis of the impact of large peer-to-peer networks on the underlying network it is meaningful to use realistic topologies with a large amount of routers and links [2], [21].

In accordance with real-world network structures, here topologies which are hierarchically structured and based on power law graphs [22] are used. A topology is represented as a graph $G(V, E)$ which contains sets of vertices's and edges. In the Internet context the vertex is a router with properties like capacity and location. A edge is a link with the bandwidth and a start- and end router. All links of a graph are per default bidirectional. Thus the links (edges) become duplicated to unidirectional back- and forward-edges. Optionally, we can define the bandwidth for every direction separately. Each node has a fixed geographical location and for one and only node. If there are several nodes at one place there they are aggregated into one node.

Generally, we use a typical Internet topology at the Autonomous System (AS) level that contains three layers, a backbone, several regions and at the lowest level the access network respectively the LANs. The LAN structures are not mapped in an absolutely exact way, because the end-systems are connected directly with the access router (Point-of-Present, PoP) of the backbone. This abstraction is taken since the distances in the LAN are quite small compared to the distances in the backbone.

There are two address spaces, one for the physical network structure and the second for the overlay network. To a physical node in the network more than one overlay node (resp. a application end-system) can become allocated. Thus, the ALS has an address system analogical to the real-world with overlay address and TCP/IP address space. This differentiation is necessary to model real-world behavior. For example, weeks after the turn-off of our experimental peer-to-peer system, a considerable amount of traffic, addressed to this peer-to-peer system, was still measurable.

As already described in Section 2, both ALS and the ScenGen packet level simulation are based on an identical topology. So we can use the results from the ALS as input for the ScenGen simulations. With the exact traffic model for the application layer, the ALS system is used for traffic generation at packet level. In the following Table I, a example for this output data is given.

TABLE I
INTERCHANGE DATA OF ALS AND SCENGEN

stime	packet	snode	intermediate	enode
...				
0.0728907	LOGIN(52)	45	13,2,3,6	25
0.0895437	CONNECT(52)	25	6,3,1,12	58
0.0923754	ACK(52)	58	12,1,3,6	25
0.0943661	LOGIN(52)	51	10,1,2,13	45
0.0963656	LOGIN(52)	45	13,2,1,10	51
0.101625	CONNECT(52)	51	10,1,3,6	25
...				

stime: start time in minutes, *packet:* message type and size, *snode:* physical address of start node, *intermediate:* comma separated list of intermediate nodes, *enode:* end node

By means of a graph model, which is based on the Boost Graph Library [20], all graphical tasks are computed such as the routing in a realistic network. The shortest path routing (similar to the prevalent Open Shortest Path First, OSPF) to model a realistic routing behavior is used. The routing information and the graph structure are basis for the traffic forwarding in the next Section.

3.2 Traffic Forwarding

The Traffic Forwarding describes the transport of data from the source to the destination over a communication network. The main property is the duration of a transmission, i.e. the end-to-end delay. A good overview on modeling the end-to-end (e2e) delay can be found in [23], [24]. ALS applies an empirical model for the e2e delay. In the next paragraph it will be discussed.

In current peer-to-peer networks, several millions of users can be active simultaneously. Packet-layer simulations of such large and complex systems are limited by the performance. The difficulties in simulating large communication networks are discussed in studies [25], [17]. Therefore, in this approach a upper abstraction level is applied and consider only the application messages. Depending on the peer-to-peer protocol, the size and the content of a message is given. In the next step we are interested in the duration of the transfer of the message from the source to the destination peer. So the relevant property of a transmission in a communication network is in our approach the end-to-end delay.

There are many factors which influence the end-to-end delay. Considering all these factors (e.g. background traffic resp. noise, packet loss, etc.) could result in a suboptimal solution since to handle so many complex and difficult parameters consequently prohibits scalability. Thus we pursue the idea of using measurements as statistical pattern for the end-to-end delay. In the following, the modeling of the traffic forwarding in our simulation is described.

The end-to-end delay between the peers P_1 and P_2 must determined, see Figure 2. The dashed line in Fig. 2 represents the end-

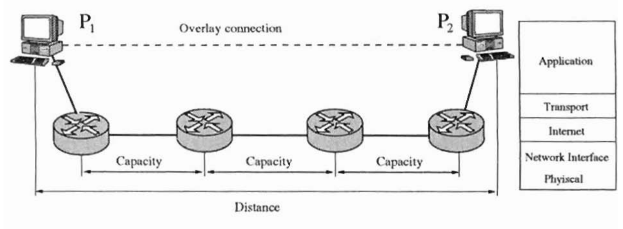

Figure 2 – Example scenario for a traffic forwarding in a peer-to-peer overlay network

to-end delay between both end-systems. All the traversed links and routers are known. From this the end-to-end delay between two communicating peers is estimated. Messages from the peers incurring transmission delay T_h, the queuing delay Q_h, processing delay S_h and propagation delay P_h at each hop h from the source to the destination. Thus we get

$$D = \sum_{h \in Path} (T_h + Q_h + S_h + P_h) \qquad (1)$$

The only random component of the delay equation (1) consists of the queuing delay in the network, $Q = \sum_{h \in Path} Q_h$. The value of the total delay depends on the number of intermediate nodes (routers). In the example of Figure 2 we have five hops and four intermediate routers. The deterministic part of the transmission over the five links can be determined by the message size, the distance and the electromagnetic travel time in through the physical path. In [26], [27] the authors propose several distributions to determine the e2e delay that are based on measurement results in the Internet. Thus we are able to determine a realistic e2e delay with the suggested distributions and depending from the traversed route in the network. Later on, there is the possibility to verify the results in the ScenGen packet level simulation and if necessary restart the ALS with new parameters. At the moment the different e2e delays in the simulations is a problem. The estimation of the e2e delay delivers other results than the fairly exact computation of ScenGen, but the deviations are quite small.

3.3 User and Data Model

Our user model describes the behavior of a user who uses a peer-to-peer client software. We use the notations "peer client", "peer" and "user" as synonyms because in this model a user can only start one client and a client corresponds to a peer. The data model represents the resources of a peer-to-peer network such the probability of the resource sizes. Both models are interdependent because in a peer-to-peer system the behavior of the user is based on the search of resources. At first we describe the user model and hereafter the data model.

A typical action of a peer-to-peer user is to connect with the peer-to-peer network. In the next step he can start to search for resources or stay online and the peer-to-peer client is able to process requests from other clients. After a certain duration the user leaves the peer-to-peer system. The user behavior depends on the peer-to-peer system (see the protocol Section 3.4), the daytime and many more parameters. There are many real-world observations and analysis of peer-to-peer traffic characteristics [28], [29], [30] that deal with these peer-to-peer parameters and distributions. ALS models a peer-to-peer user as an exponential ON/OFF source. Thereby the ON state is again divided in two sub-states : the *ACTIVE* state and the *IDLE* state. In the ACTIVE state the peer-to-peer client is currently sending a request to the peer-to-peer network. Otherwise the client is in the in the IDLE state. Thus the client is ready to process queries from the other peers. For changing between both states the Pareto or Exponential distributions are used. The distribution and the parameter depends on the used peer-to-peer protocol (see Section 3.4). The next important property of the user behavior is the mean upstream and downstream bandwidth of a peer-to-peer client. As well distributions based on the specific protocol [21], [31] are used.

Further we describe the data model which characterize the size and the rank of the shared resources. For the distribution of the size of files we can use some measurements for example [30], [31]. In the first step we apply a log-normal distribution for the determination of the file sizes like in [32]. But this approach is not exact and does not fit to each peer-to-peer system. For example the author of the measurement study presented in [33] argue that KaZaA client users share more video data than the eDonkey users. So the file size distribution of both peer-to-peer systems are quite different. The second item of the data model is the rank of a file. Based on the observation that only a few files produce the majority of the traffic volume the choice of the shared files have a big influence on the underlying Internet. If a peer starts to send a request, first by the distribution laws the rank and the size of the searched file will be determined. A possibility for a distribution is Zipf's law [21]. Then the request will be sent to adjacent peers. By the rank of the requested file every peer can determine the chance of success for an incoming request. The requesting peer gets messages from each peer who can provide the searched resource whereby all the steps for a query in a peer-to-peer system depends on the peer-to-peer protocol we handle in the next Section.

3.4 Protocol

The last part of the ALS framework is the protocol implementation. It is a quite generic part in order to create simulations with several protocol implementations. As already described in the introduction we consider peer-to-peer systems. For the conception phase we apply an implementation of a virtual super-peer protocol as described in [34]. We use this virtual protocol approach as a first step for further development toward most popular peer-to-peer systems like Gnutella, eDonkey and KaZaA. The central task of a peer-to-peer protocol is to support the searching for resources in the peer-to-peer search (overlay) network .

Thus the protocol supports the evaluation of incoming queries and if necessary the forwarding of this requests. Also it sends the own request toward the network and checks the number of hops of the request in the network. Because a peer-to-peer system is decentral organized, the protocol is responsible for the maintenance of the structure of the search network. For example in a super-peer structure many peers are connected with one super-peer. If too many peers are connected with one super-peer the system has to restructure itself. In a first step an additional super-peer is appointed and the peers are divided between the two super-peers. The rules for these functions are all central and must accomplish this functionality. Thus the protocol in a peer-to-peer systems has crucial influence on the whole search network and consequently to the underlying network resources. The properties of a peer-to-peer protocol are the:

- structure of the peer-to-peer search network,
- search behavior (e.g. number of hops, searching with distributed hash tables)
- network control (for example create or finish a connection, still alive messages, etc.)
- initial behavior (bootstrapping)
- influence of the user behavior
- etc.

In order to make a meaningful study about the impact of a peer-to-peer system at the underlying Internet it is very important so model

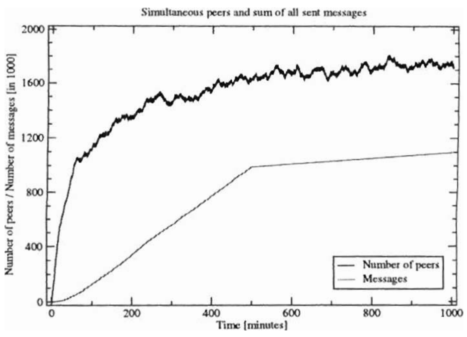

Figure 3 – ALS Simulation at bootstrap phase

the protocol as realistic as possible.

As proof of concept, Figure 3 shows an example of a simulation run using our simulation framework software. In this example a simple peer-to-peer broadcast protocol was implemented and tested. The graph depicts the bootstrap phase of the simulated peer-to-peer overlay network. The AFS results are calibrated with the packet level simulation.

4. SUMMARY AND CONCLUSIONS

This paper provides a description of our tools for application and packet level simulations. Our application level simulation (ALS) framework is specialized for analyzing P2P applications on the application level. Our approach is much more scalable than a pure network level simulation approach. ALS takes into account the physical network structure and a realistic delay distribution. It can model the characteristics of different P2P protocols. For our analyzes, we focus on super-peer applications at the moment. The result of the application level simulation are fed into a packet level simulation using our KOM ScenGen tool collection. It can also be used as traffic generator for our lab testbed consisting of 24 FreeBSD routers. This way the ALS results can be verified on a subnetwork and certain network parameters like loss and queuing delay can be measured more exactly than with a pure application level experiment.

A big challenge is to model peer-to-peer systems in our simulation environment. Consequently, we plan to implement several more realistic peer-to-peer protocol. We are able to generate reproducible results at the application and packet level. Goal is to analyze the impact of peer-to-peer traffic of the subnetwork of a ISP. We are convinced that traffic from overlay networks like peer-to-peer have strong effects to the network planning in the future.

Another important challenge is the enhancement of the peer-to-peer systems. There the issue to enable interworking of the peer-to-peer protocols with the underlying Internet. Further, to create a IP friendly overlay protocol. This does not any apply for peer-to-peer systems but other overlay approaches like GRID, too. Our simulation tool can help analyzing the effects of these effects.

REFERENCES

[1] O. Heckmann, K. Pandit, J. Schmitt, and R. Steinmetz, "KOM ScenGen - The Swiss Army Knife For Simulations and Emulation Experiments," *MIPS*, 2003, recipient of the Best Paper Award.

[2] V. Paxon and S. Floyd, "Why We Don't Know How To Simulate The Internet," in *In Proceedings of the Winder Communication Conference*, December 1997. [Online]. Available: citeseer.nj.nec.com/paxon99why.html

[3] "P2P Journal - P2P Simulator," 2003, http://www.p2pjournal.com/. [Online]. Available: http://www.p2pjournal.com/

[4] "Network Simulator NS2," http://www.isi.edu/nsnam/ns/.

[5] "JavaSim Network Simulator," http://www.javasim.org/.

[6] "OpNet Network Simulator," http://www.opnet.com/.

[7] O. Heckmann, "Topologies for ISP level network simulation (website)," http://www.kom.e-technik.tu-darmstadt.de/ heckmann/topologies/.

[8] TIERS., "Tiers Topology Generator," 2003. [Online]. Available: http://www.isi.edu/nsnam/ns/ns-topology.html#tiers/

[9] BRITE, "Boston University Representative Internet Topology Generator," http://www.cs.bu.edu/brite/.

[10] A. Medina, I. M. Lakhina, and J. Byers, "BRITE: Universal Topology Generation from a User's Perspective," 2001, http://www.cs.bu.edu/brite/.

[11] GT-ITM, "Georgia Tech Internetwork Topology Models," http://www.cc.gatech.edu/projects/gtitm/.

[12] "Inet Topology Generator," http://topology.eecs.umich.edu/inet/.

[13] G. Kramer, "UC Davis Generator of Self-Similar Traffic," http://wwwcsif.cs.ucdavis.edu/˜kramer/code/trf_gen2.html.

[14] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," in *Proceedings of the ACM SIGCOMM*, 1998.

[15] K. Cho, "The Design and Imlementation of the AltQ Traffic Management System," Ph.D. dissertation, Keio University, January 2001.

[16] "KOM RSVP Engine," http://www.kom.tu-darmstadt.de/rsvp/.

[17] G. F. Riley and M. H. Ammar, "Simulating Large Networks: How Big is Big Enough?" *Proceedings of First International Conference on Grand Challenges for Modeling and Simulation*, Jan. 2002.

[18] P. Huang, "Enabling Large-scale Network Simulations: A Selective Abstraction Approach," *USC Computer Science Department Technical Report 99-715*, September 1999. [Online]. Available: citeseer.nj.nec.com/huang99enabling.html

[19] ComNets Lehrstuhl für Kommunikationsnetze der RWTH Aachen, "ComNets Class Library and Tools (CNCL)," http://www.comnets.rwth-aachen.de/doc/cncl.html.

[20] J. G. Siek, L.-Q. Lee, and A. Lumsdaine, *Boost Graph Library, The: User Guide and Reference Manual.* Addison-Wesley, 2001, http://www.boost.org/libs/graph/doc/.

[21] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic Across Large Networks," *Second Annual ACM Internet Measurement Workshop*, November 2002.

[22] C. R. Palmer and J. G. Steffan, "Generating Network Topologies That Obey Power Laws," in *GLOBECOM '2000*, 2000.

[23] M. J. Coates and R. D. Nowak, "Network Tomography for Internal Delay Estimation," *In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, Salt Lake City, UT*, 2001.

[24] M. Allmann and V. Paxson, "In Estimating End-to-end Network Path Propertiers," *SIGCOMM'99*, 1999.

[25] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet," *IEEE/ACM Transactions on Networking.*, 1997. [Online]. Available: Available at http://www.aciri.org/floyd/papers.html.

[26] G. Hooghiemstra and P. V. Mieghem, "Delay Distributions on Fixed Internet Paths," *Delft University of Technology, Technical Report 20011020*, 2001.

[27] C. Bovy, H. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. V. Mieghem, "Analysis of End-to-end Delay Measurements in Internet," *Proceedings of Passive and Active Measurement (PAM2002), Fort Collins, USA, March 25-27*, pp. 26–33, 2002.

[28] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-peer file sharing systems," in *Proc. of Infocom*, 2003. [Online]. Available: citeseer.nj.nec.com/560486.html

[29] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Technical Report UW-CSE-01-06-02*, 2001.

[30] N. Leibowitz, M. Ripeanu, and A. Wierzbicki, "Deconstructing the Kazaa Network," *3rd IEEE Workshop on Internet Applications (WIAPP'03), San Jose, CA, June 23-24,*, 2003.

[31] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts," *Multimedia Systems Journal, Volume 8, Issue 5*, November 2002.

[32] G. D. Costa and O. Richard, "Impact of Realistic Workload in Peer-to-Peer Systems - a Case Study Freenet," *In Proceedings of the ISPDC*, 2002.

[33] Sandvine Incorporated, "Regional characteristics of P2P - File sharing as a Multi-application, Multi-national Phenomenon," *An Industry White Paper*, 2003.

[34] B. Yang and H. Garcia-Molina, "Designing a Super-peer Network," in *19th Int'l Conf. Data Engineering, IEEE Computer Society Press, Los Alamitos, CA.*, Mar. 2003.