The Two-Step P2P Simulation Approach

A Framework for Message- and Packet-Level Simulation

Hannes Birck, Oliver Heckmann, Andreas Mauthe and Ralf Steinmetz

***Abstract*— In this article a framework is introduced that can be used to analyse the effects & 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.**

***Index Terms*— simulation, modeling, peer-to-peer overlay networks and protocols**

I. Introduction

Simulations are necessary to investigate crucial issues and possible system behavior of communication networks. A simulator has to be scalable on one side to allow the investigation of very large networks. On the other side, the simulator should also yield realistic result. This means that a good simulator reflects the network behavior realistically. Unfortunately, there is a trade-off between those two goals as modeling the network behavior realistically typically increases the computational effort and therefore decreases the scalability.

In this work we present a novel approach for an application layer simulation (ALS). It promises realistic as well as scalable network simulations using a two step approach. It 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. It is especially well suited for the currently very popular peer-to-peer applications as represented by the peer-to-peer file sharing systems (“the fastest growing Internet applications ever” [2]). These presently pose a great challenge for simulators and simulation models. Latest measurements showed that up to 80% of the traffic volume in the networks of Internet Service Provider (ISP) is generated by peer-to-peer systems [3], [4].

A further important issue 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 accuracy and the detail level of the modeling as for instance discussed in [5], [6]. 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 [7].

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. In Section II we discuss related work to this topic. Section III describes the packet-level simulation and emulation toolset. Section IV presents the novel application layer simulation approach, and Section V finished with the summary and the conclusion.

II. Related Work

In [1] the KOM ScenGen is described, a tool that supports network level simulation and testbed experiments. Among other things it allows to run the same experiment in a network simulator and independently in a testbed. In this paper a framework for application level simulations (see Section IV) is introduced that can be used to evaluate P2P protocols and

Manuscript received June 15, 2004; revised July 12, 2005 and August 12, 2005. The paper was presented in part at the Conference on Software, Telecommunications and Computer Networks (SoftCOM) 2004.

H. Birck, O. Heckmann and R. Steinmetz are with KOM Multimedia Communications Lab, Darmstadt University of Technology, 64283 Darmstadt, Germany (e-mail: {birck,heckmann}@kom.tu-darmstadt.de).

A. Mauthe is with Computing Department InfoLab 21, South Drive, Lancaster University, Lancaster LA1 4WA, United Kingdom (e-mail: andreas@comp.lancs.ac.uk).

1845-6421/05/5138

to generate realistic P2P traffic. The generated traffic can be further analysed with KOM ScenGen.

There are several existing studies about simulating and modeling of peer-to-peer systems. Reference [8] describes a quite detailed packet level simulation based on the Network Simulator (NS2) [9]. An empirical approach is taken in the work presented in [10]. The study addresses the modeling of the Freenet peer-to-peer system. Another more analytical study [11] uses the "Query-Cycle Model" to model the request for resources of a peer-to-peer system.

Much research has also been focused on the difficulties of simulating large networks. The simulation of the Internet is topic of [5], [6]. In [12] the problem of the model size in relation to the real-world is discussed. For example it is surely not useful to generate with 1000 nodes simulation model realistic results for a large network with approx. 1 millions nodes. Peer-to-peer simulators/demonstrators are presented at the P2PJournal web side [7] in the submenu "P2P simulators" and in [13]. These simulators are able to depict e.g. the structure of a peer-to-peer overlay network or show the reconstruction procedure in a hierarchical system.

Recent measurement studies are [3], [4]. These studies contain measure results of the peer-to-peer traffic volume in the ISP's networks. The studies [14]–[16] describes peer-to-peer traffic models and characteristics.

However, in all this approaches there is no direct line between the application models and the underlying network structure.

III. 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 is 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.

A. Terminology and Traffic Description

The term **traffic** is used to describe the amount of bits that is transmitted over one link or is 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. For ATM traffic we can distinguish between cell, burst and flow layer [17]. For IP traffic we found that the 5 layers of Figure 1 are appropriate.

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

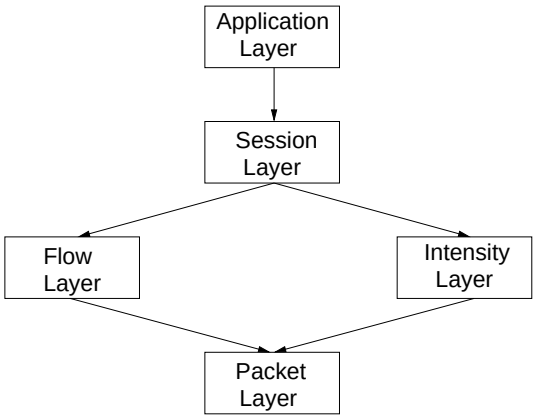

Figure 1: Traffic Layers

are an example that typically uses 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
- TCP algorithm parameters

A CBR flow for example is characterized additionally by the following parameters:

- End time
- Packet size
- Interval between two packets

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 is 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 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 [9], JavaSim [18], OpNet [19] etc. are used for network simulation. Our presented approach currently uses NS2 for packet level simulations and our own framework for the application level simulations.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.

B. Conducting an Experiment

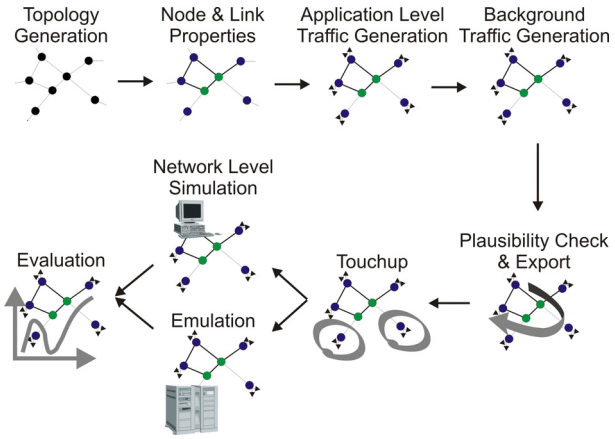

Figure 2: Conducting a Network Experiment

Figure 2 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 [20] and a converter for different topology generators like TIERS [21], BRITE [22], [23], GT-ITM [24] and Inet [25]. Topologies can also be created with a special GUI. See also Section IV-A.

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, for details see [26].

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

- Delay of a link
- Bandwidth of a link
- Queuing algorithm and queue size of a link
- Traffic properties of a node
  The traffic properties of a node specify the type of traffic this node is producing or consuming. For each type of traffic a model is specified. The specified traffic model is used in a later step to create and describe arbitrary traffic.

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** 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 [27].

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[1] loss probabilities while the other one is more sophisticated and based on M/M/1 queues and the TCP formula [28]. 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 that sets up the topology and the traffic sources and starts them. In a second OTcl file the user has the opportunity to finetune the setup process for his needs. For more details see [29].

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 is 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 [30] and dummynet [30] 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

[1]Estimated by the experimenter.

for the traffic emulator tool written introduced in [31] are distributed to all nodes and can be started automatically. The 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.

IV. 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 [5], [6], [12], [32]. 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) [33]. CNCL is an object oriented library for event driven simulations. For graphical task the Boost Graph Library (BGL) [34] 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.

A. 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 IV-D. 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 IV-B paragraph.

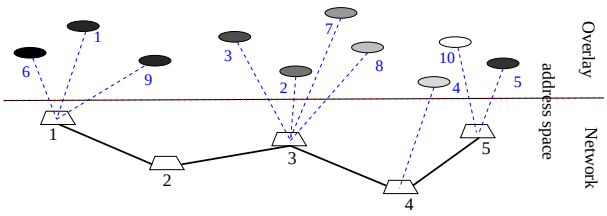

Figure 3: Mapping of the overlay structure to the network structure

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 [7], [13]. 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 [6], [35].

In accordance with real-world network structures, here topologies which are hierarchically structured and based on power law graphs [26], [36], [37] 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. An 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.

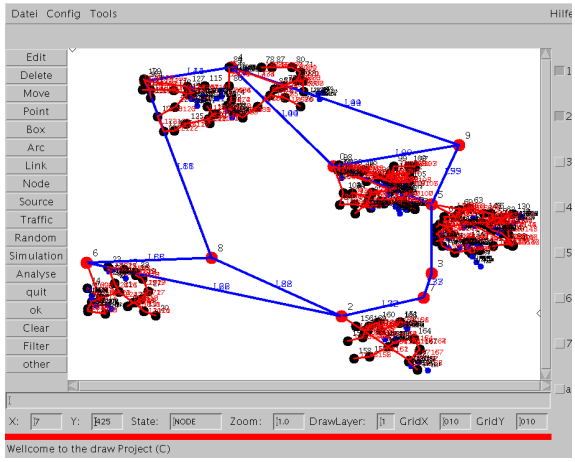

Figure 4: The Scenario Editor for the AFS framework

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. an application end-system) can become allocated, see Figure 3. 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.

The ALS framework is able to import several topology formats. This enables to use topology generators like BRITE [23], TIERS [21], etc. Using topology generators, it is possible to create realistic topologies with an arbitrary amount of

routers and different structures. A good description for the generation of realistic topologies is [26]. For case studies and experiments with special configurations the AFS frameworks supports a Scenario Editor (Fig.4) for this purpose. With this editor the properties of the network structure and nodes resp. peers are modifiable. Thus it is possible to create special configurations for experiments at the application layer.

Additional the ALS framework provides the possibility to create output data for the Network Animator (Nam) of the Network Simulator packet [9]. With this visualization tool the dynamical behavior of peer-to-peer networks are very well visual demonstrable.

As already described in Section III, 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, an 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
0.102479	CONNECT(52)	51	10,1,12	58
0.103461	ACK(52)	25	6,3,1,10	51
0.103732	ACK(52)	58	12,1,10	51
0.145715	LOGIN(52)	42	9,2,13	45
0.146783	LOGIN(52)	45	13,2,9	42
0.150255	CONNECT(52)	42	9,2,1,10	51
...				

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 [34], 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 important for the traffic forwarding that is descripted in the next section.

B. 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 [38]–[42]. 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 [6], [12]. Therefore, in this approach an 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 in our overlay network analysis approach the relevant property of a transmission in a communication network is 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 5. The dashed line in Fig. 5 repre-

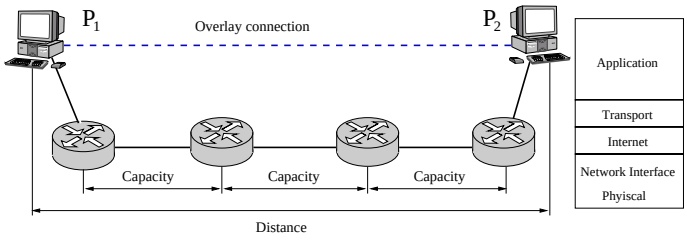

Figure 5: Example scenario for a traffic forwarding in a peer-to-peer overlay network

sents the end-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 5 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 [43], [44] the authors propose several distributions to determine the e2e delay that is 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.

C. User and Data Model

Our user model describes the behavior of an user who uses a peer-to-peer client software. We use the notations “peer client”, “peer” and “user” as synonyms because in this model an 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 certain duration the user leaves the peer-to-peer system. The user behavior depends on the peer-to-peer system (see the protocol Section IV-D), the daytime and many more parameters. There are many real-world observations and analysis of peer-to-peer traffic characteristics [15], [16], [45], [46] that deal with these peer-to-peer parameters and distributions. ALS models a peer-to-peer user as an exponential ON/OFF source as depicted in Figure 6. Thereby the ON state is again divided in two sub-states (see lower part of Fig. 6): 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 IV-D). 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 specific measurements and analyses (see [35], [47]) are used.

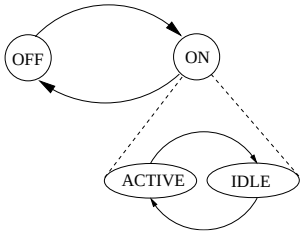

Figure 6: States of the peer-to-peer user model

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 [46], [47]. In the first step we apply a log-normal distribution for the determination of the file sizes like in [10]. 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 [4] argues that KaZaA client users share more video data than the eDonkey users. So the file size distribution of both peer-to-peer systems is 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 has 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 [35]. 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.

D. 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 comparison and analyzes of hybrid unstructured peer-to-peer networks we apply an implementation of a virtual super-peer protocol as described in [48]. This *Multi-Layer-Overlay-Protocol* (**MLOP**) is a prototype lightweight Overlay Protocol to test and analyze hierarchical structures in dynamic overlay networks. Our MLOP should not be another approach to solve or optimize special overlay protocol problems, but it is an approach to analyze and compare hybrid unstructured peer-to-peer networks in dynamical network environments. Thus, it is possible to understand and explain the success of systems such as KaZaA, eDonkey or Gnutella. An important point is the possibility to explore the effect of hierarchically. In the following we give a short description of MLOP.

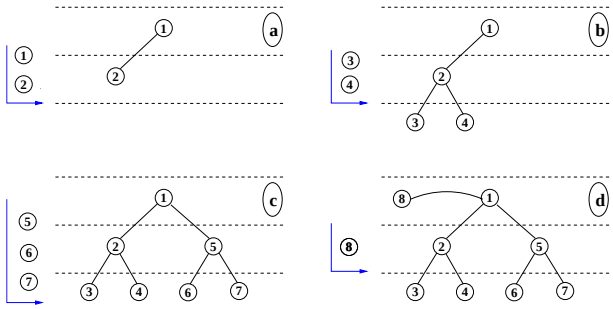

Figure 7: Example of the construction of the MLOP overlay network structure

MLOP is derived from the Open Fasttrack Protocol (see for instance [49], [50]) a clone of the Fasttrack Protocol of the well known KaZaA Client. In [51] we have analysed the OpenFastTrack protocol, the parameter and the behavior of other clients at our request. We also have kept statistics about the distribution of e.g. packet size, answer behavior and message delay.

The main idea of MLOP is to “keep it simple” in contrast to the quite complex openFT/Fasttrack protocol. Thus the

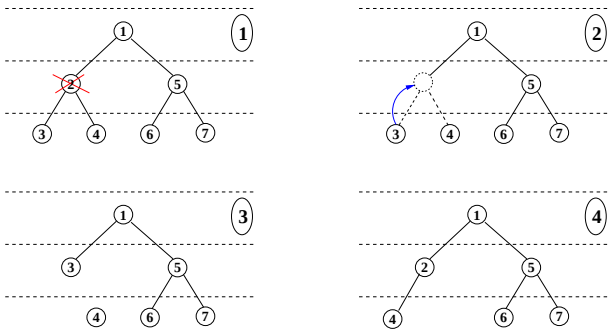

Figure 8: Example of the de- and reconstruction process in MLOP

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 decentralise organized, the protocol is responsible for the maintenance of the structure of the search network.

Any joining node first has to contact the login server to get an assignment of the level and the list of the other nodes. Therefore the login server has a very strong influence on the obtained network structure because here the level of every node is assigned. The actual algorithm provides a complete fill of each network layer starting from the top-level. Figure 7 depicts an example of the construction of a 3-layer network with a node degree of 2. Starting from the top-level first the leaf-level is filled with nodes. If the subtree is completely filled a new top-level node is created (see figure 7.d).

To keep the predefined structure the nodes have to check periodically the relations to their neighbors and/or parents. For this purpose a node sends a PING-message after a timeout interval (default is 120 seconds) to each of their neighbors or to their parent node. When the node does not receive a PONG-message as answer, the corresponding parent or neighbor will remove. The node is forced to establish new relationships to

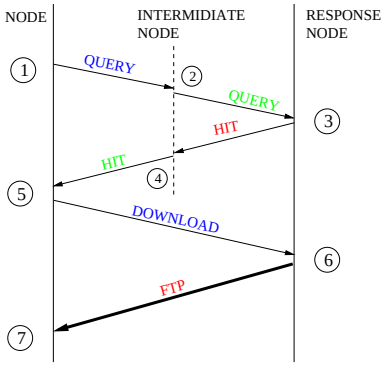

Figure 9: Diagram of the retrieval process

other nodes. When a node notices that his parent node is not longer reachable it tries to reconnect to the overlay network:

1) send a REJECT-message to all of his sons (when existing).
2) start to get a new parent/neighbor node and if this attempt unsuccessful the login server will contacted with a GET_LIST-message. So the node will receive an actual list with participating nodes from the login server and tries again to get a new parent or neighbor.

Figure 9 shows the reconstruction process of a MLOP overlay network after the leave of node 2 in the first picture of the sequence. Also here the MLOP follows the idea “as simple as possible”, because the protocol does not attempt to maintain the structure of a subgraph. The subgraph will decompose and every node tries to reconnect to the overlay network.

MLOP supports the retrieval and exchange of arbitrary resources, such as audio or video data files.

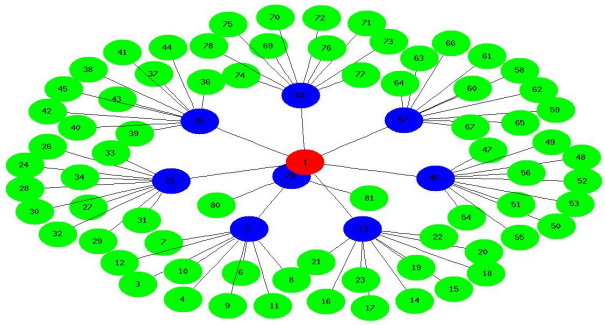

Figure 10: An example of a 3-layer MLOP graph

In figure 9 the MLOP query process is plotted. A retrieval for a specific resource starts with a QUERY-message to each of his neighbors or to his parent (figure 9 see step 1). When a node receives a QUERY-message either the node finds the resource in his own content or he forwarded this QUERY-message to his parent or neighbors (step 2). In step 3 a node has found the demanded resource and answers with a HIT-message that contains the IP address of this node. This HIT-message will forward at exact the same path that has taken the QUERY-message to this node, so the privacy of the requesting node is sustained. When the requesting node receives the HIT-

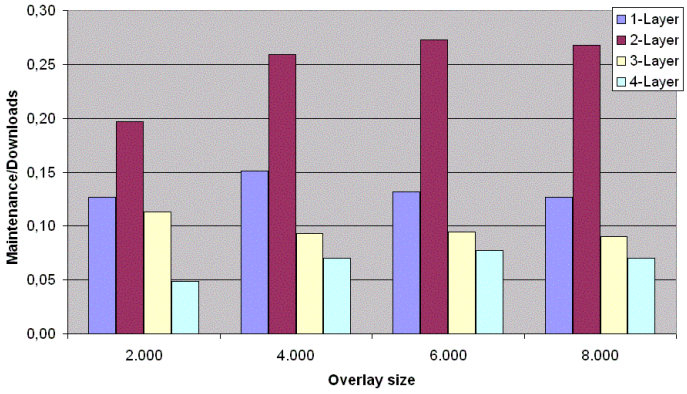

Figure 11: Efficiency factor of several overlay architectures

message he can send a DOWNLOAD request and receives the demanded resource via FTP.

The complete description of MLOP would go beyond the scope of this article and so we continue with testbed results of ALS and MLOP. In our experiments we have extended the

Table II: Comparison of overlay architectures

Layer	Number of nodes	Signalling traffic	Efficiency factor
1	2.000	5.246.769	0,1266
2	2.000	2.764.147	0,1971
3	2.000	3.723.188	0,1135
4	2.000	5.484.132	0,0490
1	4.000	4.300.365	0,1507
2	4.000	4.869.016	0,2590
3	4.000	8.516.109	0,0932
4	4.000	9.466.787	0,0701
1	6.000	4.586.461	0,1318
2	6.000	7.044.447	0,2727
3	6.000	12.430.989	0,0947
4	6.000	13.491.292	0,0773
1	8.000	6.767.196	0,1268
2	8.000	9.277.783	0,2676
3	8.000	16.465.328	0,0905
4	8.000	18.260.360	0,0701

super-peer concept. While in the original concept only two layers exist; a top layer with the super peers and an underlying layer with the nodes connected to the superpeers. We have enhanced this concept with arbitrary additional layers. In the con- and destruction examples in figure 7 and 8 a 3-layer structure is used.

In the following we present an experiment with ALS and MLOP. In this experiment several hierarchical architectures are compared. As metric to compare the efficiency of architectures we use the ratio of the maintenance costs and the success rate of a retrieval process. Thus, the results define the number of signalling traffic per retrieval hit. Table II and figure 11 shows the results of an experiment with four different architectures with 1, 2, 3 and 4-layers and four different number of nodes in the overlay network. The greater values for the efficiency factor are the better results. The diagram depicts the advantage of 2-layer architectures for this small overlay networks. A 2-layer architecture is more than two times more effective than a “flat” 1-layer architecture for a MLOP overlay network with maximal 6000 nodes. Our experiment shows that the protocol in a peer-to-peer system may has crucial influence on the whole overlay network and consequently to the underlying network (bandwidth) resources. 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 the protocol as realistic as possible.

V. Summary and Conclusions

This paper provides a novel approach for simulating large networks. Our approach is scalable and at the same time yields realistic and trustworthy results. Our approach and the developed tools are suited for application and packet level simulations. Our application level simulation (ALS) framework is specialized for analyzing P2P applications on the application level. It 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 results 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.

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.

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] C-NET NEWS, “Napster among fasted-growing Net technologies,” *Internet*, Oct. 2000.

[3] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, and T. Seely, “Packet-Level Traffic Measurements from the Sprint IP Backbone,” *IEEE Networks*, pp. 6–15, 2003.

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

[5] 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

[6] 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.

[7] “P2P Journal - p2p-simulator.org,” 2005. [Online]. Available: http://www.p2pjournal.com/

[8] Q. He, M. Ammar, G. Riley, H. Raj, and R. Fujimoto, “Mapping Peer Behavior to Packet-level Details: A Framework for Packet-level Simulation of Peer-to-Peer Systems,” in *In Proceedings of MASCOTS*, 2003.

[9] “Network Simulator NS2,” http://www.isi.edu/nsnam/ns/.

[10] 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.

[11] M. T. Schlosser, T. E. Condie, and S. D. Kamvar, “Simulating a P2P File-Sharing Network,” in *First Workshop on Semantics in Peer-to-Peer and Grid Computing at the Twelfth International World Wide Web Conference*, 2003.

[12] 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.

[13] N. S. Ting, “A Generic Peer-to-Peer Network Simulator,” in *Proceedings of the 2002-2003 Grad Symposiun, Computer Science Department, University of Saskatchewan*, 10 April 2003.

[14] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, “Measurement, modeling, and analysis of a peer-to-peer file-sharing workload,” in *In Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-19), Bolton Landing, NY.*, October 2003.

[15] 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

[16] R. Schollmeier and A. Dumanois, “Peer-to-Peer Traffic Characteristics,” *In Proceedings of the 9th EUNICE Open European Summer School (EUNICE’03) . Budapest-Balatonfüred, Hungary,*, September 8-10, 2003.

[17] J. Roberts, U. Mocci, and J. V. (Eds), *Broadband Network Teletraffic (Final Report of COST 242).* Springer Verlag LNCS 1155, 1996.

[18] “JavaSim Network Simulator,” http://www.javasim.org/.

[19] “OpNet Network Simulator,” http://www.opnet.com/.

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

[21] TIERS., "Tiers Topology Generator," 2003. [Online]. Available: http://www.isi.edu/nsnam/ns/ns-topology.html#tiers/
[22] BRITE, "Boston University Representative Internet Topology Generator," http://www.cs.bu.edu/brite/.
[23] A. Medina, I. M. Lakhina, and J. Byers, "BRITE: Universal Topology Generation from a User's Perspective," 2001, http://www.cs.bu.edu/brite/.
[24] GT-ITM, "Georgia Tech Internetwork Topology Models," http://www.cc.gatech.edu/projects/gtitm/.
[25] "Inet Topology Generator," http://topology.eecs.umich.edu/inet/.
[26] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, "On Realistic Network Toplogies for Simulation," in *In Proceedings of ACM SIGCOMM MoMeTools, Karlsruhe*, 2003.
[27] G. Kramer, "UC Davis Generator of Self-Similar Traffic," http://wwwcsif.cs.ucdavis.edu/~kramer/code/trf_gen2.html.
[28] 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.
[29] O. Heckmann, K. Pandit, J. Schmitt, M. Hoffmann, and M. Jobmann, "LETSQoS Milestone 2," http://www.letsqos.de, June 2002.
[30] K. Cho, "The Design and Imlementation of the AltQ Traffic Management System," Ph.D. dissertation, Keio University, January 2001.
[31] "KOM RSVP Engine," http://www.kom.tu-darmstadt.de/rsvp/.
[32] 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
[33] ComNets Lehrstuhl für Kommunikationsnetze der RWTH Aachen, "ComNets Class Library and Tools (CNCL)," http://www.comnets.rwth-aachen.de/doc/cncl.html.
[34] 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/.
[35] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic Across Large Networks," *Second Annual ACM Internet Measurement Workshop*, November 2002.
[36] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos, "Power-Laws and the AS-level Internet Topology," in *Transactions on Networking*. IEEE/ACM, Aug. 2003, pp. 514–524.
[37] C. R. Palmer and J. G. Steffan, "Generating Network Topologies That Obey Power Laws," in *GLOBECOM '2000*, 2000.
[38] D. Yates, J. Kurose, D. Towsley, and M. G. Hluchyj, "On Per-Session End-to-End Delay Distributions and the Call Admission Problem for Real-Time Applications with QoS Requirements," *SIGCOMM'93 - Ithaca*, 1993.
[39] 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.
[40] M. Allmann and V. Paxson, "In Estimating End-to-end Network Path Propertiers," *SIGCOMM'99*, 1999.
[41] H. Ohsaki, M. Murata, and H. Miyahara, "Modeling End-to-End Packet Delay Dynamics of the Internet Using System Identification," *Presented at International Teletrac Congress*, 17, Sept. 2001.
[42] J.-C. Bolot, "Characterizing End-to-End Packet Delay and Loss in the Internet," *Journal of High Speed Networks, Vol. 2, Issue 3*, 1993.
[43] G. Hooghiemstra and P. V. Mieghem, "Delay Distributions on Fixed Internet Paths," *Delft University of Technology, Technical Report 20011020*, 2001.
[44] 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.
[45] S. Saroiu, P. K. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," in *SPIE Multimedia Computing and Networking (MMCN2002)*, 2002, http://www.cs.washington.edu/homes/gribble/papers/mmcn.pdf.
[46] 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.
[47] 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.
[48] 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.
[49] "gift project website," 2005. [Online]. Available: http://sourceforge.net/projects/gift
[50] "gift :: Openft network snapshots," 2005. [Online]. Available: http://fnord.csbnet.se/openft/
[51] T. Combé and H. Birck, "Analysis of the FastTrack Protocol," *Studentsproject at the Darmstadt University of Technology*, 2005.

Hannes Birck received his Diploma degree in computer science from the Darmstadt University of Technology, Germany, in 1997, and is currently working toward the Ph.D. degree. From 1998 to 2001, working on applied research projects in traffic engineering and network planning at the Deutsche Telekom Research Centre in Darmstadt. In 2002 he changed to the computer centre of the Darmstadt University of Technology, as a member of the management board for the acquisition of new telecommunication equipment. Since November 2002, he has hold a position as a Research Assistant at the Multimedia Communications Lab (KOM), aiming at a Ph.D. thesis. His particular research interests include Networks Analysis, Distributed Systems, Peer-to-Peer Networking and Network Simulations.

Oliver Heckmann finished his Ph.D. in computer science with the title "A System-oriented Approach to Efficiency and Quality of Service for Internet Service Providers" in 2004. Since then he is research group leader of the Multimedia Distribution & Networking groups of the Multimedia Communications Lab (KOM) at the Darmstadt University of Technology. He is working for the eFinance Lab (www.efinancelab.de) and the Internet"okonomie (www.internetoekonomie.uni-frankfurt.de) research projects. From 2000 to 2004 he was a researcher at the same lab working in the M3I (www.m3i.org) and letsqos (www.letsqos.de) research projects. His research areas are Internet Service Providers, IT Architectures and Peer-to-Peer Networking. His special focus is quality of service, efficiency and scalability in these environments.

Andreas Mauthe is a Senior Lecturer at the Computing Department, Lancaster University. He has been working in the area of distributed and multimedia systems for more than 15 years. His particular interests are in the area of content management systems and content networks, large scale distributed systems, Peer-to-Peer systems, and self-organisation aspects. Prior to joining Lancaster University, Andreas was heading a research group at the Multimedia Communications Lab (KOM) of the Technical University of Darmstadt. After completing his PhD in Lancaster in 1997, Andreas worked for more than four years in different positions in industry. He was General Manager UK Operations, Chief Development Officer (CDO) and member of the division's management board of the Content Management Systems Division of Tecmath AG (now Blue Order), a German based software house and system integrator working in the area of content management (mainly for the broadcast industry). Andreas has been acting as organiser, chair, and programme committee member for various conferences. Further, he has been participating in standardisation activities (e.g. ISO and SMPTE) and served as expert advisor and evaluator for the European Commission.