Dominik Stingl, Christian Groß, Julius Rückert, Leonhard Nobach, Aleksandra Kovacevic, Ralf Steinmetz: *PeerfactSim.KOM: A Simulation Framework for Peer-to-Peer Systems.* In: Waleed W. Smari (Edit): Proceedings of the 2011 International Conference on High Performance Computing & Simulation (HPCS 2011), p. 577-584, IEEE, July 2011. ISBN 978-1-61284-382-7.

PeerfactSim.KOM: A Simulation Framework for Peer-to-Peer Systems

Dominik Stingl, Christian Gross, Julius Rückert, Leonhard Nobach, Aleksandra Kovacevic, Ralf Steinmetz
Multimedia Communications Lab (KOM), Technische Universität Darmstadt
Rundeturmstr 10, 64283 Darmstadt, Germany
E-mail: {stingl,gross,rueckert,nobach,sandra,steinmetz}@kom.tu-darmstadt.de

***Abstract*—Since P2P systems have become popular in the late nineties, simulation of these systems has always been a preferable method of performance evaluation. Simulations facilitate the development and evaluation of new protocols and mechanisms, while enabling a comparison of existing solutions. In this paper, we present PeerfactSim.KOM, a discrete-event P2P simulator that is suitable for a wide range of varying scenarios in the area of P2P. It consists of a layered architecture, provides a broad selection of P2P protocols for the modeled layers, and eases the implementation of new components through its modular design. In addition, the simulator provides helpful tools to configure and evaluate a simulation scenario.**

1. Introduction

During the last decade, the Peer-to-Peer (P2P) paradigm attracted many researchers from Computer Science and motivated them to research and investigate new concepts and mechanisms for P2P systems. Apart from the challenge to design and implement new ideas, the process to prove, evaluate, and compare one or several approaches states a problem on its own. Naicken et al. [19] identified that the P2P research community applies three different techniques for performance evaluation to tackle the analysis of one or several solutions. The mentioned evaluation techniques cover analytical modeling, simulations, and experiments. Based on the characteristics of these evaluation techniques, which were summarized and compared by Jain [11], simulations turned out to be the most popular and adequate tool to evaluate and test concepts in the area of P2P, as observed in [20]. The reasons for the popularity of simulations might result from the simplification and unification in analytical modeling, while the execution of experiments in testbeds suffers scalability, reproducibility and requires existing prototypes.

With PeerfactSim.KOM, we present our solution for a P2P simulator that tackles the previously sketched shortcomings of analytical modeling and experiments. The simulation framework already provides a variety of P2P protocols addressing overlays, decentralized services, and applications, which can be used to implement, test, or compare new protocols. Besides its suitability for a wide range of scenarios, the simulator includes a separate logging and statistics architecture that alleviates the crucial process of data capturing of ongoing simulations, as stated by Naicken et al. [20]. Furthermore, the integrated visualization allows to illustrate and debug the communication process of executed simulations. Besides these features, PeerfactSim.KOM addresses the general requirements of P2P simulators, as identified by [3] and [20]. Therefore, the simulator consists of a modular and flexible architecture with a set of interfaces for the different components. Below the P2P-relevant layers, the simulator comprises an underlying network model to simulate the transmission of data. Besides the modular architecture, PeerfactSim.KOM focuses on scalability to execute simulations in a reasonable amount of time, while preserving computational resources.

The next section of this paper details the related work of P2P simulators. Afterwards, Section 3 describes the modular architecture and underlying concepts of the simulator, whereas Section 4 copes with the options that a user has before, during and after a simulation. Section 5 gives insights about the performance of our simulation framework, while the last section presents the conclusion of the paper.

2. Related Work

The popularity of simulation as evaluation technique has resulted in a vast amount of overlay and network simulators. Network simulators, such as NS-3[1], simulate network protocols and examine the protocols' characteristics through realistic models, covering aspects such as network topology or congestion. For the evaluation and analysis of P2P systems, the sketched network models are out of scope, because they come with inherent computational costs, hindering the simulation of P2P mechanisms at a larger scale. Therefore, as identified by Naicken et al. [19], the majority of P2P simulators that consists of a layered architecture, which is oriented at the ISO-OSI model, abstracts the lower layers of networks or completely omits them. In the following, we describe a small fraction of existing P2P simulators with their underlying concepts. For this description, we subdivide the contemplated tools in two classes. The first class contains the simulators that are limited to simulations.

PeerSim [18], which constitutes a prominent example of the first class, is a Java-based simulator with a modular architecture to simulate different P2P mechanisms on top of an underlying configurable network model. The available P2P mechanisms range from overlay protocols to algorithms for aggregation or management. PeerSim offers a discrete-event engine to simulate experiments with detailed protocols (e.g. for

[1] http://www.nsnam.org/

the underlying network) and a cycle-based engine to simulate simplified models at a larger scale.

In contrast, *PlanetSim* [22] is a pure discrete-event simulator, which is also written in Java. It consists of three different layers, that model the network, the overlay and the application. For the implementation of new protocols, the simulator offers two different approaches. The algorithm-based approach implements the complete functionality of a protocol within a single component, while the behavior-based approach divides the separable aspects of a protocol and implements them in separate components.

The second class of P2P simulators addresses and supports the development process from initial simulation models of a mechanism to prototypical implementations for real networks. A simulator of this class targets the complete re-usage of code without modifications for real experiments in testbeds by replacing the underlying network model and the event-driven simulation engine with a real network interface and timers.

ProtoPeer [6] is a Java-based simulator, where a user can switch between simulations and real world experiments to evaluate his P2P system. The layered architecture of ProtoPeer can be subdivided into three layers. The layer on top includes the components for an application and the P2P protocol. The layer below consists of the networking and time APIs, which are implemented through the components of the lowest layer. This layer either models the network or constitutes the network interface with real timers for network experiments.

OverSim [3] states a further approach that belongs to the second class of P2P simulators but that is written in C++. The architecture of the simulator consists as well of three layers, covering an application, overlay and network layer. The overlay and network layer offer interfaces, such as a KBR [5] or UDP interface, to the layers above and allow for a transparent exchange of the implementing components. Just as in ProtoPeer, the user can choose between different models for the network layer or replace the model with a network interface to execute real world experiments.

3. Simulator Architecture

PeerfactSim.KOM is a Java-based simulator for investigating large-scale P2P systems. The architecture of the simulator comprises a discrete-event simulation engine that manages the simulated peers, which communicate with each other by exchanging messages. The main objective of the simulator is to be applicable to different use cases and to facilitate the simulation of a wide range of varying scenarios in the area of P2P. Therefore, the simulator consists of a layered architecture that tries to cover the diverse aspects of a P2P system through the provided layers, as depicted in Figure 1.

For each of the depicted layers, one or more interfaces exist that offer their functionality to the remaining layers. The interfaces alleviate the development of new components for a layer and facilitate the exchange of a component with other implementations. Based on these interfaces, the simulator provides the concept of *default* and *skeletal implementations*. A default implementation represents an implementation of a component, whose offered functionality can be explicitly defined and implemented, and which may serve other simulations without modifications (e.g. the component for the event scheduler). On the contrary, if the functionality of a component cannot be clearly identified, the simulator offers a skeletal implementation that can be extended by the user. The skeletal implementation already includes an abstract implementation base, such as the integration in the simulator and the interaction with other simulation components, relieving the user from these tasks.

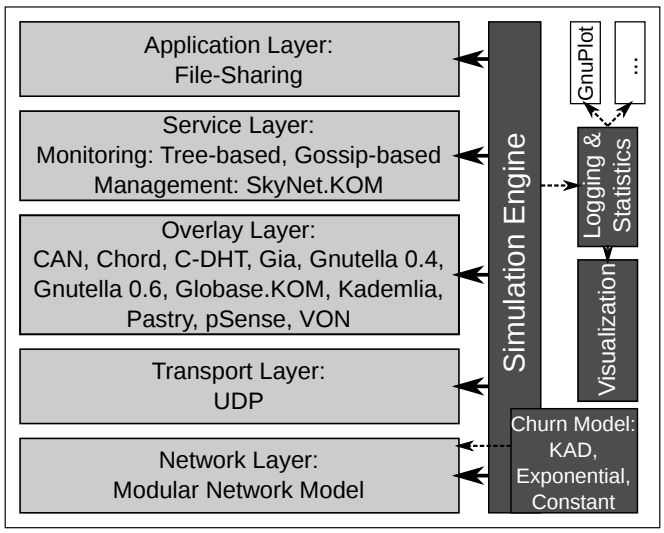


Figure 1. The Modular Architecture Of PeerfactSim.KOM

In the following subsections, we describe the underlying concepts for each of the five layers shown in Figure 1 and list existing implementations for them. Section 4 subsequently covers the remaining layers.

A. Network Layer

PeerfactSim.KOM focuses on the simulation of P2P systems. Therefore, it does not model the underlying network topology for data transmission between peers. Instead, the network model applies mathematical functions as well as data from Internet measurement projects (e.g. CAIDA[2] or PingER[3]) to simulate data transmission between arbitrary peers. Depending on the level of detail, the model can include influencing factors for a transmission, such as latency, jitter, or peer positioning.

Our network model, forming the basis for the sketched approach, consists of two components, the *Network Layer* and the *Subnet*. A separate instance of the Network Layer is installed at every simulated peer and is responsible for sending and receiving messages from the modeled network and for passing them to the Transport Layer above. The additional component Subnet abstracts the network and simulates the transmission of data through the network. In contrast to the

[2]http://www.caida.org/projects/macroscopic
[3]http://www-iepm.slac.stanford.edu/pinger

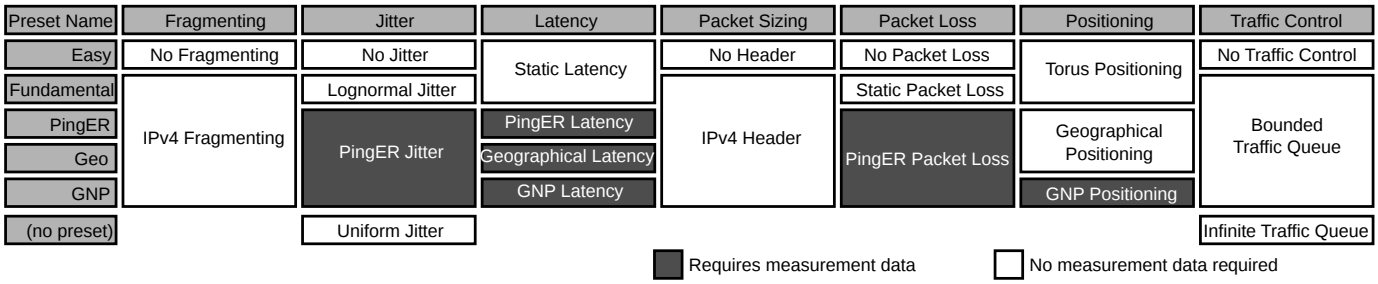


Figure 2. An Overview Of Different Strategies, Grouped Into Different Strategy Types (Columns) And Recommended Settings (Rows)

numerous instances of the Network Layer, which comply with the number of simulated peers, a simulation just comprises one instance of the Subnet, which connects all peers with each other. For these two components that form the basis of our network model, PeerfactSim.KOM provides skeletal implementations, which can be used as starting point for user-specific network models.

At the moment, PeerfactSim.KOM provides a default network model for simulations, comprising an implementation for the Network Layer and the Subnet. Instead of creating multiple implementations for different network models with varying levels of detail, we tried to identify and separate the important aspects of a network model, which affect the transmission of data during a simulation, and which can be independently modeled with varying levels of detail. Based on the identified aspects, we developed the Modular Network Model, which consists of the separable components called *strategies*. With the described Modular Network Model, we aim at providing a network model, which allows for a flexible configuration of the Network Layer and the Subnet, and which comes with a comfortable extendability for future strategies.

The identified strategies of the modular model cover different aspects for the transmission of data. The Subnet comprehends the *Latency* and *Jitter* strategy to model the varying latency of data transmission between two peers, as proposed by Kaune et al. [13]. The *Packet Loss* strategy calculates the probability that a sent message gets lost. In addition, the *Fragmenting* strategy specifies how a message is fragmented, whereas the *Packet Sizing* strategy determines how the size of a message is calculated. For the sender and receiver, the Modular Network Model offers two strategies that comprise the peer position (*Positioning* strategy) and how sending and receiving of a message is processed at a peer (*Traffic Control* strategy).

For the mentioned strategies, PeerfactSim.KOM provides the implementations, which are displayed in Figure 2, and which can be independently chosen or selected through presets to configure the Modular Network Model. The included strategies comprise simple concepts based on probability functions (e.g. *Lognormal Jitter* strategy) or on static functions, for example adding a constant amount of time to every data transmission. On the contrary, the Modular Network Model offers more realistic strategies, which rely on data from Internet measurement studies. The *PingER Jitter*, *PingER Latency*, and *PingER Packet Loss* strategies apply measurement data from the PingER project to implement the strategies. The calculation of the latency in the *GNP Latency* strategy, employing the measurement data from the CAIDA project, follows the approach proposed by Kunzmann et al. [16]. The strategy is based on an euclidean embedding of the peers in a multidimensional space, using the technique of Global Network Positioning (GNP) [21].

In order to model the heterogeneity of the simulated network devices, the Network Layer consists of an additional component, which is separated from the chosen network model and specifies the characteristics of the simulated network devices. Currently, PeerfactSim.KOM offers two implementations for this component to assign different upload and download capacities to the network devices. The first approach randomly chooses a value from a defined interval to determine the network device capacities. The second approach selects the network devices and their capacities based on a report from the OECD [1].

B. Transport Layer

The main task of the Transport Layer in PeerfactSim.KOM is to provide an end-to-end communication service to higher layers, including P2P overlays or applications. Since the Transport Layer, just as the Network Layer, belongs to the condensed lower layers of a P2P simulator, the detailed mechanisms provided by an ordinary Transport Layer (e.g. connection-oriented data streams or flow control) are most of the time not the focus of P2P simulations. The implementation of this layer, therefore, abstracts over the correct implementation of most services and mainly supports the transmission of UDP-messages. For future versions, we plan to integrate a model that simulates simplified TCP for the transmission of data.

C. Overlay Layer

The Overlay Layer represents an important layer in PeerfactSim.KOM, because it contains the implementations of the different P2P overlay models. Based on these overlays, diverse applications or additional distributed services (e.g. application layer multicast or monitoring) can be set up. In the following, we list the existing overlays in PeerfactSim.KOM and divide them into different classes. Currently, the bunch of overlays can be classified into unstructured and structured overlays as

Table 1
IMPLEMENTED P2P OVERLAYS

Overlay Class	Implementation
Unstructured	Gnutella 0.4 [2], Gnutella 0.6 [14], Gia [4]
Hybrid	Globase.KOM [15]
Structured	CAN (2-dim.) [23], Chord [27], C-DHT, Kademlia [17], Pastry [24]
IDOs	VON [10], pSense [25]

well as into overlays for spatial information dissemination. While the use cases for the first two classes of overlays are already well known, the latter class can support Network Virtual Environments (NVE) as known from Massively Multiplayer Online Games (MMOG). Table 1 lists the implemented overlays and groups them into the aforementioned three classes.

Instead of detailing the implemented overlays in our simulator, we focus on the given structure and underlying concepts of the layer. Due to the varying functionality of overlays, we refrain from designing an interface that incorporates all methods of an overlay node. Instead, we provide a class hierarchy, allowing the developer to choose, which functionality the own overlay node should at least provide (see Figure 3). Based on the sparse `OverlayNode`-interface that defines how the overlay has to be integrated in the simulator, the extending `JoinLeaveOverlayNode`-interface defines methods for joining and leaving an overlay. At the moment, this class hierarchy only provides an additional marker-interface for unstructured overlays as well as two interfaces for structured overlays. The latter two interfaces address the common functionality of structured overlays, as outlined by Dabek et al. [5]: The capabilities of Key-Based Routing (KBR) are defined in the `KBR`-interface, while `DHTNode` integrates the functionality of Distributed Hash Tables (DHT). In addition, PeerfactSim.KOM offers further components that many overlay protocols comprise and that we consider to be important. These range from an overlay routing table, over the bootstrapping mechanism to the overlay ID and key. Besides the interfaces, the Overlay Layer includes two skeletal implementations, which shall relieve the developer from integrating a new overlay into the layered architecture and the simulator. These skeletal implementations comprise the abstract definition of a node within the overlay as well as a base message type that can be used as basis for further messages of the overlay protocol.

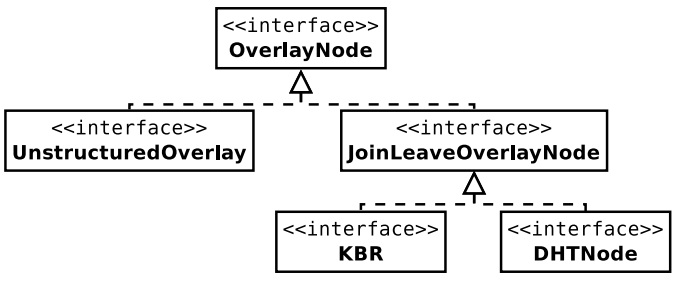


Figure 3. Class Diagram For The Functionality Of A Peer Within A P2P Overlay

D. Service Layer

The Service Layer is intended for components that offer additional services to an application or to the whole system. The services can range from application layer multicast over publish/subscribe mechanisms to monitoring and management approaches. Like overlays, the services of the Service Layer mainly operate in a decentralized fashion and depend on the underlying overlay to provide the additional functionality. For this reason, we decided to place them as an external layer between the Application and Overlay Layer. To implement new services, we use the same basic building blocks as for the Overlay Layer.

Currently, the Service Layer contains the tree-based monitoring solution SkyEye.KOM proposed by Graffi et al. [7], which is set up on top of DHTs. It provides a global view for a predefined set of attributes, which are monitored at different layers of the P2P system. In addition, the monitoring mechanism is used in the management framework SkyNet.KOM [8] that maintains and improves the underlying overlays based on the collected data and preset quality intervals. The second monitoring solution in the simulator was presented by Jelasity et al. [12] and uses a gossip-based approach to monitor the attributes from the P2P system.

E. Application Layer

On top of the layered architecture, the Application Layer can host P2P applications. Currently, PeerfactSim.KOM only provides a file-sharing application for the Application Layer. During the simulation of the application, the peers can publish their files and seek further files of other peers based on a given probability distribution. The application currently runs on Kademlia and Chord as well as on Gnutella 0.6 and Gia. The evaluation of the simulator presented in Section 5 defines a scenario with this application on top of Kademlia and Gnutella 0.6.

4. PEERFACTSIM.KOM IN USE

Having detailed the layered architecture and the modular design of PeerfactSim.KOM, this section covers the creation of scenarios in the following subsection. Afterwards, we examine the logging and statistics architecture for the produced data of a simulation (see Subsection 4-2) and finally highlight the visualization of a simulated experiment and its statistics in Subsection 4-3.

A. Running A Simulation

For the creation of a simulation in PeerfactSim.KOM, a XML-based configuration file is used. This file denotes, which layers are included, which implementation represents a single layer (e.g. Chord for the Overlay Layer), and how they are configured (e.g. the *GNP*-preset for the Network Layer). Moreover, the general setup comprises the number of simulated peers, the duration of a simulation, the responsible classes for collecting data, the churn generator, and an action file. The action file is written in a script-like language and specifies when a certain action should be executed by a host

or a group of hosts. If required by the executed method, several parameters can be passed.

To model the dynamics of the whole system comprising the autonomic arrival and departure of peers, PeerfactSim.KOM provides a churn generator. Based on a mathematical function, the generator chooses a peer and determines the next point in time when the peer will go on- or offline. The churn generator works at the Network Layer of a peer and connects or disconnects the layer of a peer with or from the Subnet. Consequently, the Overlay Layer of that peer is responsible to join the existing overlay during the connection establishment, while the rest of the peers must handle the ungraceful departure of that peer, when it is disconnected. We denote the time between arrival and departure as *session time*, while the time between departure and arrival is denoted as *intersession time*. Currently, PeerfactSim.KOM supports three churn models: (i) the constant model provides a static session and intersession time for each peer, (ii) the exponential model uses an exponential distribution to calculate the session and intersession time, and (iii) the KAD model is based on a Weibull distribution that was derived from measurements in a real KAD overlay by Steiner et al. [26].

B. Logging And Statistics Architecture

PeerfactSim.KOM provides its own architecture for gathering data of ongoing simulations. The architecture can be divided into a logging and a statistics part. While logging is mainly used to trace and debug a simulation, the simulator offers the statistics architecture to grab the important data for on-the-fly statistics or for later post-processing. It is not possible to predict all kinds of data that might be important for the evaluation. Hence, we integrate a separate statistics architecture in the simulator that addresses the collection of basic data types from which composite data types can be derived. The core of the monitoring architecture consists of two elements: A system-wide Monitor (available as default implementation) and the `Analyzer`-interface. To collect data from a simulation, the simulator defines a set of interception points that are used by the Monitor to grab the data. A user who is interested in some specific type of data, implements the Analyzer or one of its nested interfaces and registers it at the Monitor, which in turn is responsible for the notification of an Analyzer depending on the type of monitored data. Besides the basic Analyzer that only defines methods for starting and stopping an analyzer, nested analyzers for specific components exist. For example, they can monitor data from the Network Layer, from the Churn Generator or from KBR-based overlays.

C. Visualizing A Simulation

After the execution of a simulation with PeerfactSim.KOM, the simulator offers the possibility to visualize the executed simulation. The integrated visualization component, as displayed in Figure 4, allows for the visualization of the topology and the exchanged messages of the simulated P2P system. Dealing with the presentation of the topology, the visualization can organize the peers based on the provided coordinates of the Network Layer or arrange them in a ring-like topology. Through the varying thickness of a displayed peer or an edge between two peers, metrics, such as the amount of neighbours or of transmitted messages, can be represented.

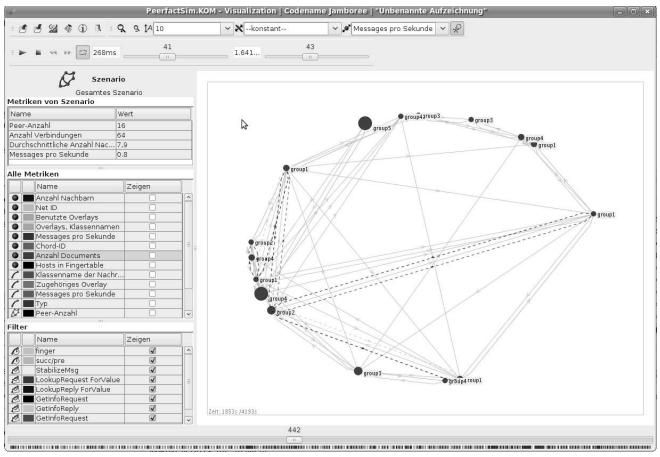


Figure 4. The Visualization Component Of The Simulator

5. Performance Evaluation

Within this section, we examine the performance of PeerfactSim.KOM for simulating different P2P overlays on top of the provided underlay models. Regarding the evaluation of the performance, we measure the simulation duration and the maximum consumed amount of memory. In addition, we show that simulation time and memory consumption do not necessarily depend on the amount of transmitted messages, but on the complexity of the respective protocol that runs on each peer.

We simulate a file-sharing application on top of Gnutella 0.6 and Kademlia. While Gnutella 0.6 is an unstructured P2P overlay, which uses a simple but robust gossip-based communication protocol, Kademlia is a structured P2P overlay, which applies sophisticated mechanisms to manage routing tables and to replicate the sharable objects. The general simulation setup is displayed in Table 2. Dealing with the scenario, the peers are equally divided into four groups, which successively join the overlay during the first 80 minutes. Afterwards, each peer publishes its files before the random lookup for a file starts. The latter two actions are only executed by half of the peers, while the deployment of churn, starting after two hours, is applied to all peers.

Every simulation was run five times, each time with a different seed. The graphs, depicted in Figure 5, display the mean and the 95% confidence interval for each metric. The described scenarios are simulated on a server with Ubuntu 10.04 (64bit) and the JDK 6. The hardware consists of four processors (two Dual-Core AMD Opteron Processors 2214) with 2200MHz and 64GiB of main memory. The maximum amount of memory that a simulation may consume is limited to 20GiB.

Considering the simulation duration of the four scenarios

Table 2
SIMULATION SETUP

Simulation Component		Setup	
Modular Network Model	Easy-, GNP-preset	Churn generator	Exponential model
P2P overlay	Gnutella 0.6, Kademlia	Simulation duration	180min
Application	File-Sharing	Number of peers	500, 1,000, 5,000, 10,000, 50,000

depicted in Figure 5(a), we observe an increase of the simulation duration that exceeds a linear growth, irrespective of the chosen P2P overlay or network preset. The evaluation of the impact of different underlay presets on the simulation duration outlines that simulations, using the Easy-preset, are faster than simulations with the detailed GNP-preset. The increased simulation duration mainly results from the additionally scheduled events for the simulation of a transmission between two peers. The influence of the P2P overlay on the simulation duration does not only depend on the message overhead of a protocol, but on its computational complexity simulated at each participating peer. While the produced message overhead of Gnutella always exceeds the one of Kademlia (Figure 5(c)), larger simulations with Kademlia (>5,000 peers) are more time consuming, due to the higher complexity of the protocol, which becomes the predominant factor for the simulation duration.

The memory consumption of the simulator resembles the trend of the simulation duration. Figure 5(b) outlines the linear increase of the memory consumption with the number of peers. In this regard, smaller simulations (<1,000 peers), which use an underlay preset that requires measurement data, e.g. GNP, are an exception, because the memory consumption is mainly influenced by loading the required data for the underlay model. Dealing with the impact of overlays on the consumed memory, it becomes apparent that the message overhead of an overlay is not necessarily the predominant factor for the memory consumption. Instead, comparing 5(b) and 5(c), we observe for the comparison of different overlays that state-intensive and complex protocols mainly account for the consumed memory of larger simulations with PeerfactSim.KOM.

6. CONCLUSION AND FUTURE WORK

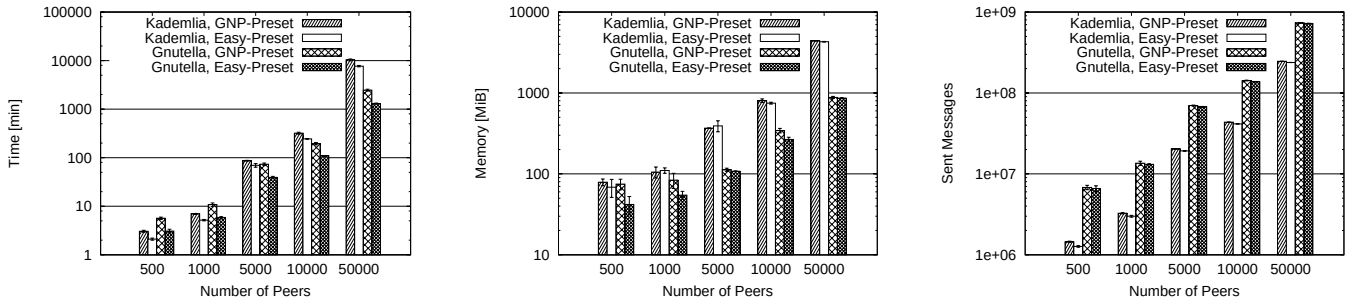
In this paper, we presented PeerfactSim.KOM, a simulation framework for P2P systems, which already offers existing implementations for a variety of P2P protocols. The simulator consists of a modular architecture that can be subdivided into components. The functionality of these components is defined by interfaces to alleviate the development of new solutions and to enable the exchange of different implementations for one component. Besides the architecture, we introduced flexible tools to configure simulation setups and to create scenarios. Using the provided logging and statistics architecture, simulations can be easily debugged, while important information and statistics of a simulation are collected. The additional visualization component of PeerfactSim.KOM illustrates an executed simulation. Regarding the performance and scalability of PeerfactSim.KOM, our evaluation shows that the simulator allows to execute experiments with tens of thousands of peers in a reasonable amount of time, while minimizing the memory consumption. Moreover, the presented analysis sketches the influence of chosen components, such as overlay or underlay presets, on the simulation duration and memory consumption. For the future, we plan to extend the Transport Layer with an efficient model for TCP. Furthermore, we want to test and improve the interaction with the Common Simulator Interface [9] and examine its applicability with new implementations of existing P2P overlay protocols. Information about PeerfactSim.KOM and its source code are available at `http://www.peerfactsim.com`.

ACKNOWLEDGEMENTS

This work has been supported in parts by the German Research Foundation, Research Group 733, “QuaP2P: Quality Improvement of Peer-to-Peer Systems” and by the German Federal Ministry of Education and Research (BMBF) in the Project “Premium Services” (support code 01IA08003A) and in the Project “G-Lab VirtuRAMA” (support code 01BK0920).

REFERENCES

[1] “OECD Broadband Portal” http://oecd.org/sti/ict/broadband.
[2] “The Annotated Gnutella Protocol Specification v0.4” http://rfc-gnutella.sourceforge.net/developer/stable/index.html.
[3] I. Baumgart, B. Heep, and S. Krause, “OverSim: A Flexible Overlay Network Simulation Framework” in *IEEE Global Internet Symposium*, 2007, pp. 79–84.
[4] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, “Making Gnutella-like P2P Systems Scalable” in *Proc. of the Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2003, pp. 407–418.
[5] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, “Towards a Common API for Structured Peer-to-Peer Overlays” in *Proc. of the 2nd Int. Workshop on Peer-to-Peer Systems*, 2003, pp. 33–44.
[6] W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer, “ProtoPeer: A P2P Toolkit Bridging the Gap Between Simulation and live Deployement” in *Proc. of the 2nd Int. Conf. on Simulation Tools and Techniques*, 2009, pp. 1–9.
[7] K. Graffi, A. Kovacevic, S. Xiao, and R. Steinmetz, “SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems” in *Proc. of the 14th Int. Conf. on Parallel and Distributed Systems*, 2008, pp. 279–286.
[8] K. Graffi, D. Stingl, J. Rueckert, A. Kovacevic, and R. Steinmetz, “Monitoring and Management of Structured Peer-to-Peer Systems” in *Proc. of the 9th Int. Conf. on Peer-to-Peer Computing*, 2009, pp. 311–320.
[9] C. Groß, M. Lehn, D. Stingl, A. Kovacevic, A. Buchmann, and R. Steinmetz, “Towards a Common Interface for Overlay Network Simulators” in *Proc. of the 16th Int. Conf. on Parallel and Distributed Systems*, 2010, pp. 27–34.
[10] S.-Y. Hu, J.-f. Chen, and T.-H. Chen, “VON: A Scalable Peer-to-Peer Network for Virtual Environments” *IEEE Network*, vol. 20, 2006, pp. 22–31.
[11] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc, 1991.
[12] M. Jelasity, A. Montresor, and O. Babaoglu, “Gossip-Based Aggregation in Large Dynamic Networks” *ACM Transactions on Computer Systems*, vol. 23, 2005, pp. 219–252.



(a) Simulation Duration (b) Memory Consumption (c) Message Overhead

Figure 5. Performance Evaluation Of PeerfactSim.KOM

[13] S. Kaune, K. Pussep, C. Leng, A. Kovacevic, G. Tyson, and R. Steinmetz, "Modelling the Internet Delay Space Based on Geographical Locations" in *Proc. of the 17th Euromicro Int. Conf. on Parallel, Distributed and Network-based Processing*, 2009, pp. 301–310.

[14] T. Klingberg and R. Manfredi, "Gnutella 0.6" http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[15] A. Kovacevic, N. Liebau, and R. Steinmetz, "Globase.KOM - A P2P Overlay for Fully Retrievable Location-based Search" in *Proc. of the 7th Int. Conf. on Peer-to-Peer Computing*, 2007, pp. 87–96.

[16] G. Kunzmann, R. Nagel, T. Hossfeld, A. Binzenhofer, and K. Eger, "Efficient Simulation of Large-Scale P2P Networks: Modeling Network Transmission Times" in *Proc. of the 15th Euromicro Int. Conf. on Parallel, Distributed, and Network-based Processing*, 2007, pp. 475–481.

[17] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric" in *Proc. of the 1st Int. Workshop on Peer-to-Peer Systems*, 2002, pp. 53–65.

[18] A. Montresor and M. Jelasity, "PeerSim: A Scalable P2P Simulator" in *Proc. of the 9th Int. Conf. on Peer-to-Peer Computing*, 2009, pp. 99 –100.

[19] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai, "A Survey of Peer-to-Peer Network Simulators" in *Proc. of The 7th Annual Postgraduate Symposium*, 2006.

[20] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers, "The State of Peer-to-Peer Simulators and Simulations" *SIGCOMM Computer Communication Review*, vol. 37, 2007, pp. 95–98.

[21] T. S. E. Ng and H. Zhang, "Towards global network positioning" in *Proc. of the 1st ACM SIGCOMM Workshop on Internet Measurement*, 2001, pp. 25–29.

[22] J. Pujol Ahullo and P. Garcia Lopez, "PlanetSim: An Extensible Simulation Tool for Peer-to-Peer Networks and Services" in *Proc. of the 9th Int. Conf. on Peer-to-Peer Computing*, 2009, pp. 85–86.

[23] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network" in *Proc. of the Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001, pp. 161–172.

[24] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-scale Peer-to-Peer Systems" in *Proc. of the IFIP/ACM Int. Conf. on Distributed Systems Platforms*, 2001, pp. 329–350.

[25] A. Schmieg, M. Stieler, S. Jeckel, P. Kabus, B. Kemme, and A. Buchmann, "pSense - Maintaining a Dynamic Localized Peer-to-Peer Structure for Position Based Multicast in Games" in *Proc. of the 8th Int. Conf. on Peer-to-Peer Computing*, 2008, pp. 247–256.

[26] M. Steiner, T. En-Najjary, and E. W. Biersack, "Analyzing Peer Behavior in KAD" pp. 1–28, 2007.

[27] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications" in *Proc. of the Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001, pp. 149–160.