OverSim: A Flexible Overlay Network Simulation Framework

Ingmar Baumgart, Bernhard Heep, Stephan Krause
Institute of Telematics
Universität Karlsruhe (TH)
Zirkel 2, D–76128 Karlsruhe, Germany
Email: {baumgart, heep, stkrause}@tm.uka.de

***Abstract*— A fundamental problem in studying peer-to-peer networks is the evaluation of new protocols. This paper presents *OverSim*, a flexible overlay network simulation framework based on OMNeT++. It was designed to fulfill a number of requirements that have been partially neglected by existing simulation frameworks. OverSim includes several structured and unstructured peer-to-peer protocols like Chord, Kademlia and Gia. These protocol implementations can be used for both simulation as well as real world networks. To facilitate the implementation of additional protocols and to make them more comparable OverSim provides several common functions like a generic lookup mechanism for structured peer-to-peer networks and an RPC interface. Several exchangeable underlay network models allow to simulate complex heterogeneous underlay networks as well as simplified networks for large-scale simulations. We show that with OverSim simulations of overlay networks with up to 100,000 nodes are feasible.**

I. Introduction

Nowadays the analysis of overlay protocols is an important task, because peer-to-peer applications are widely used and deployed. In the *ScaleNet* [1] project several academic research groups and industry partners are working on next generation networks focusing on fixed-mobile convergence and fast service deployment. Our goal in ScaleNet is to study how overlay networks can support such fast and cost efficient deployment of new services.

Next generation networks comprise heterogeneous access networks and a high mobility rate of user terminals. A foreseen feature of these networks could be the support of overlay networks by dedicated devices in the access networks to alleviate user terminals from overlay traffic. Existing peer-to-peer simulators do not support simulation of networks with these special properties. They also exhibit several major drawbacks described below. Therefore, we decided to develop a more flexible simulation framework called *OverSim* based on the discrete event simulation system OMNeT++ [2].

The rest of the paper is organized as follows: In section II we take a closer look at existing peer-to-peer simulators to avoid their shortcomings in our simulator. In section III we compile a list of requirements that we think are essential for a new approach. The detailed design of our simulation framework is described in section IV. In section V we evaluate the performance of OverSim.

A validation of the first simulation results is shown in section VI. Finally we present our ideas for enhancements and possible next steps for discussion in section VII followed by the conclusion in section VIII.

II. Related Work

There are several peer-to-peer simulators available:

P2PSim [3] is a discrete event simulator for structured overlay networks written in C++. It comes with seven peer-to-peer protocols implemented including the more recent protocols Koorde [4] and Kademlia [5]. There is a number of different underlaying network models, all of them, however, on a rather abstract level of detail, making it hard to simulate the dedicated overlay devices in the access networks mentioned above. P2PSim is largely undocumented and therefore hard to extend.

OverlayWeaver [6] is a peer-to-peer overlay construction toolkit written in Java which can be used for easy development and testing of new overlay protocols and applications. The toolkit contains a so-called *Distributed Environment Emulator* which invokes and hosts multiple instances of Java applications on a single computer. This allows the simulation of up to 4,000 nodes. Since simulations have to be run in real-time and there is no statistical output, its use as an overlay network simulator is very limited.

PlanetSim [7] is an object-oriented simulation framework for overlay networks and services written in Java. It has a well-structured and modular architecture and makes use of the Common API [8]. In addition to the overlay protocols Chord [9] and Symphony [10] there are several services like CAST and DHT available on application layer. PlanetSim offers only limited support to collect statistics and has a very simplified underlying network layer without consideration of bandwidth and latency costs. This makes it difficult to simulate heterogeneous access networks and terminal mobility. It is possible to visualize the overlay topology at the end of a simulation run, but there is no interactive GUI.

A more comprehensive survey of peer-to-peer network simulators can be found in [11], where the authors show that most available peer-to-peer network simulators have several major drawbacks limiting them in use for research projects.

III. Requirements

Our objective is to develop a novel overlay network simulation framework that is able to fulfill all the needs for a complete analysis of overlay networks with focus on large scale.

The requirements for such a simulation framework can be summarized as follows:

- **Scalability:** The simulator should be able to run simulations with a large number of nodes in a reasonable amount of time.
- **Flexibility:** The simulator should facilitate the simulation of both structured and unstructured overlay networks. The user should be able to specify all relevant simulation parameters in a human readable configuration file. The simulator should also be able to provide node mobility and node failure, as well as malicious behavior of nodes.
- **Underlying Network Modeling:** The underlying network model should be exchangeable. On the one hand a fully configurable network topology with realistic bandwidths, packet delays and packet losses should be provided. On the other hand there should be fast and simple alternative models for simulations with a large number of nodes.
- **Reuse of simulation code:** The provided implementations of overlay protocols should be reusable for real network applications enabling researchers to validate the simulator framework results by comparing them to the results from real-world test networks like PlanetLab. Therefore, the simulation framework should be able to handle and assemble real network packets and to communicate with other implementations of the same overlay protocol.
- **Statistics:** The simulator should be able to collect statistical data such as sent, received or forwarded network traffic per node, successful or unsuccessful packet delivery, and packet hop count. The output needs to be in a format that is easy to postprocess e.g. for generating gnuplot output.
- **Documentation:** For using and extending the simulator with e.g. new overlay protocols there should be a comprehensive user manual. The source code and the API have to be well documented.
- **Interactive Visualizer:** In order to validate and debug new or existing overlay protocols there should be a GUI, which visualizes both the topology of the underlying network and the overlay topology in a customizable way.

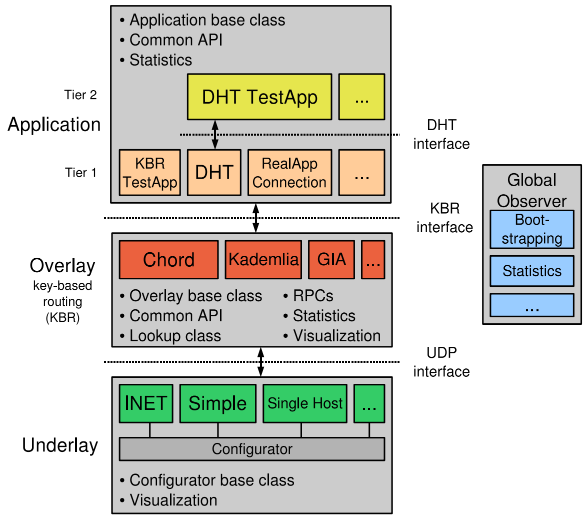

Fig. 1. Modular architecture of OverSim

IV. Design

OverSim was designed as a modular simulation framework. An overview of its architecture is illustrated in Fig. 1. In the following we describe the contained modules in detail.

A. Simulation framework OMNeT++

OverSim uses discrete event simulation (DES) to simulate exchange and processing of network messages. We make use of the open source simulation framework OMNeT++ [2], which is free for non-profit use.

OMNeT++ is highly modular. Modules are defined in a simple definition language called NED. They are either *compound modules*, which are composed from other modules, or *simple modules*, which are directly implemented in C++. These modules communicate by exchanging messages via gate connections.

To facilitate the implementation of new protocols OMNeT++ includes a message generator that generates C++ code from compact message definitions. If desired, the generated classes can be extended by the user.

OMNeT++ implements many features needed for visualization. The built-in GUI displays networks topologies, nodes and messages. Its debug features allow for deeper inspection of message contents and node variables. The GUI can be disabled for faster simulation runs.

B. Underlying network model

As postulated in section III, an overlay simulator should support different kinds of underlying network models. We implemented three network models for OverSim: *Simple*, *SingleHost*, and *INET*.

The most scalable is the *Simple* underlay. In this model data packets are sent directly from one overlay node to another by using a global routing table. Packets between overlay nodes get delayed either by a constant period of time or—for more realistic scenarios—by a delay calculated from the nodes' distance. For this, each node is placed into a two-dimensional Euclidean space. In addition the node is assigned to a logical access network characterized by bandwidth b_n, access delay d_n and packet loss, so that heterogeneous access networks can be simulated. The end-to-end packet delay d_e of packet P with length l_P between overlay nodes A and B is then calculated as follows

$$d_e = d_1 + \frac{l_P}{b_1} + c \cdot \|A - B\|_2 + d_2 + \frac{l_P}{b_2}$$

where c is constant.

This way, all relevant influences of the underlying network can be simulated with one single simulation event. Even node mobility can be achieved by changing coordinates, access network characteristics and the IP address of a node. Due to the low simulation overhead of these techniques, the *Simple* model leads to a high level of accuracy and the ability to simulate networks with a large number of nodes.

For reusing overlay protocol implementations without code modifications in real networks like PlanetLab, we developed the *SingleHost* underlay model. Here, each OverSim instance only emulates a single host, which can be connected to other instances over existing networks like the Internet.

To enable the emulated host to communicate over real networks, we implemented a new event scheduler for OMNeT++. In contrast to the default scheduler, which aims for fastest possible simulation, this scheduler slows the simulation speed down to real time. If no simulation events have to be processed, it reads data from a Linux TUN networking device that provides packet reception and transmission for user space programs. The received data packets are converted to OMNeT++ packets and then passed to the "interface card" of the simulated single node.

The *INET* underlay model is derived from the INET framework of OMNeT++, which includes simulation models of all network layers from the MAC layer onwards. It is useful for simulations of complete backbone structures, if required. Since the INET framework is not optimized for large scale networks, we had to profile the code and added hash table based routing and interface caches for faster forwarding of data packets.

With this complex underlay model we are able to analyse the benefits that can be gained by placing overlay devices into the access networks that can lead to optimization of overlay topologies and reduction of packet delay. As the INET framework contains also a model to simulate the physical layer of IEEE 802.11 networks, it can also be used to evaluate overlay protocols for ad hoc networks.

Additionally, we extended the *INET* model with similar techniques used in the *SingleHost* underlay model described above. A special "router" module can be added to the simulation scenario which acts as a gateway to a real network. This can be used to demonstrate overlay protocols with a limited number of physical devices by connecting them to a large number of overlay nodes emulated by OverSim.

All of the underlaying network models have a consistent UDP/IP interface to the overlay protocols. Hence, using a different underlaying network model is fully transparent to the overlay layer.

C. Overlay protocols

Several overlay protocols have been implemented. Most of them are structured peer-to-peer protocols like Chord [9], Kademlia [5], Koorde [4], and Broose [12], but unstructured peer-to-peer protocols like GIA [13] are available as well. In addition to these typical overlay protocols OverSim contains more specialized overlays for exchanging event messages in peer-to-peer based massively multiplayer online games like VAST [14] or the publish/subscribe-based overlay described in [15].

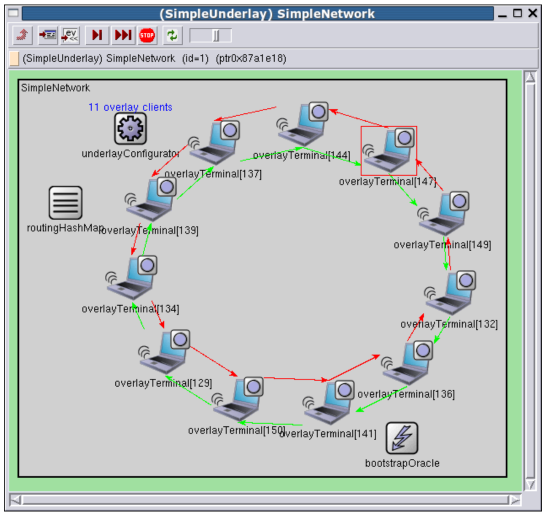

Fig. 2. OverSim showing the overlay topology of Chord

To facilitate the implementation of new overlay protocols we identified several functions that many overlay protocol implementations have in common and integrated them into our simulation framework:

- Overlay message handling (RPC and statistical data)
- Generic lookup function
- Support for visualization of the overlay topology
- Bootstrapping support

The overlay message handler provides an RPC interface to facilitate dealing with timeouts and packet retransmission due to packet losses. It also collects message related statistical data like the number of sent, received, forwarded and dropped packets per node, as well as the ratio of data to signaling traffic.

The lookup function separates the common functionality of lookup mechanisms in structured overlay networks and provides a generic iterative and recursive lookup interface. This way, for each overlay protocol only a method to query the local routing table has to be implemented, which returns the closest node in the overlay topology. The lookup function also includes basic support to simulate malicious node behaviour. The resilience of an overlay structure against attacks of malicious nodes largely depends on the number of disjoint overlay paths that exist between two arbitrary nodes. Therefore, the lookup function makes use of these redundant paths and can be used to measure the effects of some attacks on lookup success.

The graphical user interface of OMNeT++ supports debugging of new overlay protocols by showing transferred messages in detail. It is also possible to watch and even change overlay specific routing tables and other internal states during run-time. OverSim gives developers the ability to draw arrows between corresponding overlay nodes to visualize the overlay topology in addition to the underlay topology drawn by OMNeT++. This allows us to demonstrate the effects of topology adaptation mechanisms. Fig. 2 shows the visualization of the overlay topology for a Chord network.

A generic module called *Global Observer* has a global view on the overlay network in every simulation. It can fulfill several user-defined functions, but for now is mainly used as a *Bootstrap Oracle*, providing the address of a random node already in the overlay network to nodes that want to join the overlay. The module can also be used as a global statistics collector.

Sharing these mentioned common functions—which are optional to use—among different overlay protocol implementations avoids a time-consuming and error-prone rewrite of code. Besides, this helps to make overlay implementations more comparable. An example for this is that overlay protocols show the same behaviour if lookup messages have to be retransmitted due to packet loss.

The communication between overlay and application makes use of the Common API [8]. Every overlay protocol which wants to use this API has to provide at least a key-based routing interface (KBR) to the application. Overlay protocols that implement a distributed hash table (DHT) can offer this service to the application using the same interface. For protocols without such a functionality a module which suplies a common DHT can be placed between overlay and application layer using the KBR interface.

D. Applications

With the Common API design a wide range of different applications that rely on key-based routing can be evaluated with exchangeable overlays. Currently there is a KBR test application implemented, that—depending on parameter values—periodically sends test messages to random overlay keys or nodeIDs and records message delay and hop-count. More complex applications can be composed of several logical tiers.

Applications we are working on at the moment are the application layer multicast protocol *Scribe* [16] and the communication abstraction *Internet Indirection Infrastructure* [17].

V. Performance

One important requirement is the scalability of the simulator with regard to the number of nodes. We evaluated the performance by running comparable simulations

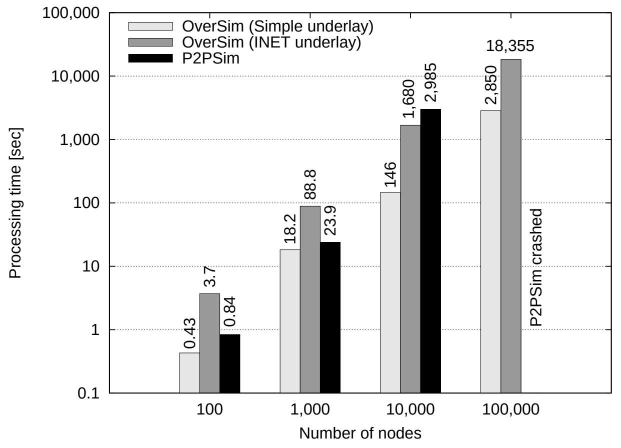

Fig. 3. Processing time as function of network size

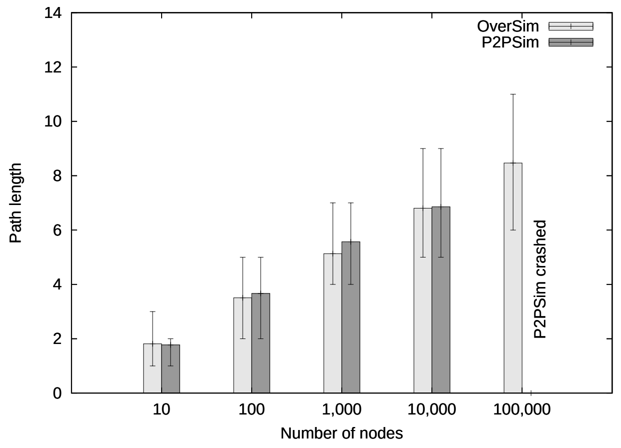

Fig. 4. Path length as function of network size

with P2PSim and OverSim and measured computation time and memory footprint. All experiments used the Chord protocol with recursive routing on an Opteron 1.8 GHz machine with 8 GB memory. After joining a fixed number of nodes we started to send messages to random destination keys.

Two different underlying network models were used for simulation runs with OverSim: *Simple* and *INET*. The *INET* model contains detailed simulations of IP queueing effects in intermediate routers on the overlay path. In our experiments we used a small, randomly connected backbone network with 10 Gbit/s links consisting of 20 backbone and 20 access routers. In contrast, the *Simple* model only added a constant delay of 50 msec between overlay nodes. The same constant delay model was also used with P2PSim.

Fig. 3 plots the processing time needed to simulate a time period of 1,000 sec dependent on the number of nodes with a logarithmic scale. In our experiments Chord's stabilize messages were sent every 20 sec, the finger table was updated every 120 sec and every 10 sec each node started a lookup for a random key.

Both simulators can be used to simulate networks of 1,000 nodes in a couple of seconds. However, when simulating a network of 10,000 nodes, P2PSim needs 45 min for a single simulation run and thus is about 20 times slower than OverSim. Our attempts to simulate a network of 100,000 nodes with P2PSim lead to a crash after running for several hours probably due to buffer overflows. In contrast, with OverSim and the *Simple* model we were able to simulate a similar run in less than an hour.

The figure also shows the effect of the level of detail of the underlaying network model on the simulator's performance. Due to the large number of events that are generated when packets travel through the network stacks of the intermediate routers, simulations with the *INET* model are about 10 times slower than simulations with the *Simple* model.

The memory consumption increases proportionally with the number of nodes. Both P2PSim and OverSim with the *Simple* model have a memory footprint of about 35 kB per node. The more complex *INET* model consumes about twice as much memory. These results show that in the investigated scenarios simulations with 100,000 nodes are very feasible with OverSim.

VI. Validation

A major problem when developing a new simulator is the validation of the implemented protocols. As a first step to validate the chord implementation we checked the correctness of the stabilization protocol by constantly performing lookups of existing nodeIDs under steady churn. The lookup results showed that a consistent ring topology was maintained on overlay layer and that the overlay correctly recovered from node failures. This was also visualized by the GUI. Furthermore, we measured the path lengths dependent on the number of nodes and compared them to the output of P2PSim. Fig. 4 shows that both simulators produce similar results. The bandwidth consumption per node for routing table maintenance and lookup traffic calculated by both simulators was also in the same order of magnitude.

VII. Future Work

At the moment we are working on the integration of several new overlay protocols and applications into the simulator. Additionally, existing implementations of overlay protocols have to be verified by comparing their simulation results with those produced by other simulators.

While our underlay models allow us to configure a number of parameters in the underlying network, there is currently no support to import models of topology generators. In order to achieve more realistic network latencies in the *Simple* model, we are planing to extend the basic synthetic coordinate model to be able to import datasets from internet latency measurements.

Although the performance evaluation showed that OverSim is quite scalable, we futher want to improve ressource usage and compare it to a broader range of simulators including PlanetSim.

VIII. Conclusion

In this paper we presented *OverSim*, a new overlay network simulation framework. The framework was developed to overcome several drawbacks of existing peer-to-peer simulators.

OverSim facilitates the implementation of new overlay protocols by providing several functions many overlay protocols have in common like overlay message handling, visualization and a generic lookup mechanism. We implemented different underlay network models to support a broad range of simulation scenarios. The *SingleHost* model e.g. allows to reuse protocol implementations in real-world networks like PlanetLab. As OverSim is based on OMNeT++, OverSim benefits from features like an efficient event scheduler and strong GUI support.

Through scalable design we have easily achieved to simulate a chord network with 100,000 nodes in a reasonable amount of time. The large number of implemented overlay protocols and the availability to collect various statistical data make OverSim a powerful tool for the peer-to-peer research community. OverSim is being actively developed as an open source project on `http://www.oversim.org/` and is open to contributions.

Acknowledgment

This research was supported by the German Federal Ministry of Education and Research as part of the *ScaleNet* project 01BU567.

References

[1] M. Siebert, B. Xu, T. Banniza, R. Keller, A. Dekorsy, J. Eichinger, R. Bless, I. Baumgart, and S. Stefanov, "ScaleNet - Converged Networks of the Future," *it - Information Technology, Themenheft "IP basierte mobile Systeme"*, vol. 5, pp. 253–263, Oct. 2006.

[2] A. Varga. Omnet++ community site. [Online]. Available: http://www.omnetpp.org/

[3] J. Li, J. Stribling, R. Morris, M. Kaashoek, and T. Gil, "A performance vs. cost framework for evaluating DHT design tradeoffs under churn," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 1, Mar. 2005, pp. 225–236.

[4] M. F. Kaashoek and D. R. Karger, "Koorde: A simple degree-optimal distributed hash table," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, vol. Volume 2735/2003, 2003, pp. 98–107.

[5] P. Maymounkov and D. Mazires, "Kademlia: A peer-to-peer information system based on the xor metric," in *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA, March 7-8, 2002. Revised Papers*, vol. Volume 2429/2002, 2002, pp. 53–65.

[6] K. Shudo. Overlay weaver. [Online]. Available: http://overlayweaver.sourceforge.net/

[7] P. Garca, C. Pairot, R. Mondjar, J. Pujol, H. Tejedor, and R. Rallo, "Planetsim: A new overlay network simulation framework," in *Software Engineering and Middleware*, vol. Volume 3437/2005, 2005, pp. 123–136.

[8] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a common api for structured peer-to-peer overlays," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, vol. Volume 2735/2003, 2003, pp. 33–44.

[9] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[10] G. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed hashing in a small world," in *4th USENIX Symposium on Internet Technologies and Systems*, 2003, pp. 127–140.

[11] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai, "A Survey of Peer-to-Peer Network Simulators," *Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK*, 2006.

[12] A.-T. Gai and L. Viennot, "Broose: a practical distributed hashtable based on the de-bruijn topology," in *Fourth International Conference on Peer-to-Peer Computing, 2004*, Aug. 2004, pp. 167–174.

[13] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like P2P systems scalable," in *SIGCOMM '03*. ACM Press, 2003, pp. 407–418.

[14] S.-Y. Hu and G.-M. Liao, "Scalable peer-to-peer networked virtual environment," in *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*. New York, NY, USA: ACM Press, 2004, pp. 129–133.

[15] S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito, "A distributed event delivery method with load balancing for mmorpg," in *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*. New York, NY, USA: ACM Press, 2005, pp. 1–8.

[16] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1489–1499, Oct. 2002.

[17] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 205–218, 2004.