HiSim: A Highly Extensible Large-Scale P2P Network Simulator

Lukas Rupprecht Jessica Smejkal Angelika Reiser
Alfons Kemper

Fakultät für Informatik, Technische Universität München
Firstname.Lastname@in.tum.de

Abstract: The popularity of Peer-to-Peer networks is increasing rapidly but developing new protocols for P2P systems is a very complex task as testing and evaluating distributed systems involves high effort. P2P simulators are being developed to tackle this difficulty, to reduce cost and to speed up development. We describe HiSim, a modular and highly scalable P2P network simulator based on the simulation framework PeerSim which we use to simulate HiSbase, a P2P framework for efficient processing of multidimensional data. Because of its modular design, HiSim can easily be extended, e.g., to fit other application domains. Basic design problems, related to query processing, are introduced in general and concretely solved within HiSim. Additionally, HiSim provides mechanisms to evaluate new protocols using an integrated statistics component. We demonstrate the high scalability by performing simulations of up to $2 \cdot 10^4$ peers and $2 \cdot 10^7$ queries.

1 Introduction

The popularity of Peer-to-Peer (P2P) networks is increasing rapidly [RD10]. Not only filesharing protocols like, e.g., BitTorrent[1] but also the science community is utilizing them heavily. *E-science* communities form data grids to process huge amounts of data (e.g. the Large Hadron Collider at Cern[2]) by combining their available resources using P2P technology.

Developing new mechanisms and protocols for P2P systems is a very complex task as it requires a distributed testing infrastructure to verify and evaluate the protocols. Such an infrastructure should be as realistic as possible but, especially in the case of large-scale P2P networks, this is hard to achieve as costs are increasing rapidly with higher numbers of peers. The distributed testing process itself is very time-consuming and complex because each system needs to be monitored individually and the gathered data needs to be combined and analyzed. Additionally, evaluating new protocols for a certain task implies a complete distributed implementation of these protocols. Sometimes this is too costly or not possible due to time limits, especially, if these protocols are only prototypes which should only give a first impression if the desired approach is working.

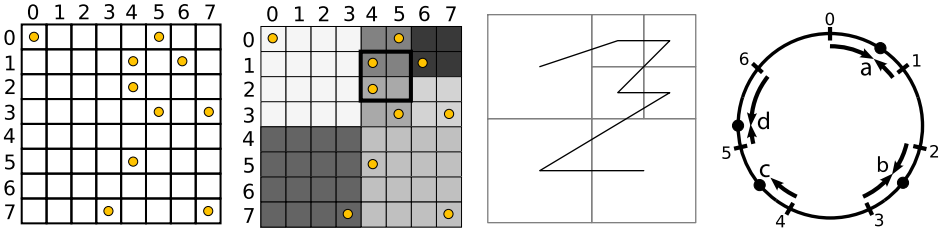
[1] www.bittorrent.com
[2] lhc.web.cern.ch/lhc

To overcome these problems, P2P network simulators are used. A simulator can run on only one single workstation and provide a central interface to attach and detach protocols easily. It allows faster and more clear testing as the whole P2P network runs centralized and the testing data does not need to be combined from several peers. In this paper we present HiSim, a highly scalable and easily extensible P2P network simulator based on the PeerSim simulation engine [JMJV], which realizes all the above mentioned advantages. We use the simulator to evaluate *HiSbase*, a P2P framework for managing and processing multidimensional E-science data. However, because of its modular design, the simulator can easily be extended to fit arbitrary application domains. The rest of the paper is structured as follows. Section 2 introduces the basics of HiSbase. Then the design of HiSim is explained. In section 4 the extensibility of the simulator is demonstrated by some examples. The next section provides a scalability analysis and sections 6 and 7 give a short overview of other simulator projects and summarize the main aspects.

2 HiSbase

HiSbase is the P2P framework for which the simulator was built. The framework provides functionality for data partitioning of multidimensional data and efficient query processing with the focus on preserving data locality and processing range queries [SBG+07], [SBM+09].

HiSbase is built upon the distributed hash table overlay structure *Pastry* [RD01]. Pastry coordinates the communication between the peers (or *nodes*) and provides a routing mechanism. It uses a one-dimensional ring topology where data and peers are mapped to. Additionally, Pastry optimizes the routing by implementing a proximity neighbor selection algorithm [CDHR03] which prefers physical neighbors when routing messages.



(a) Sample data (b) Buckets (c) Linearization (d) Peer mapping

Figure 1: A sample for data placement in HiSbase [SBG+07].

HiSbase deploys histograms to partition multidimensional data and to create buckets containing approximately the same number of data elements (see Figure 1(b)). It partitiones the data recursively, using a quadtree, and maps the leaves (or *regions*) to the overlay ring topology, using the Z-order (see Figure 1(c)). The combination of quadtrees and space-filling curves handles data skew and preserves data locality when mapping n-dimensional data to the 1-dimensional ring. Peers and regions are mapped to the key space by using

a hash function and regions are assigned to peers by their closest distance to peers (see Figure 1(d)). These histograms enable efficient query processing of region-based queries [SRK09]. Therefore a coordinating region is chosen from all regions covered by the query. A special message is routed to the node (the *coordinator*) responsible for this particular region. The coordinator then sends messages to all queried regions, and the responsible nodes look up the data in their local databases. They send the data back to the coordinator which combines all answers and sends the complete data back to the query initiator. As the space-filling curve preserves locality, it is very likely that only few nodes are responsible for the whole area covered by the query and that these nodes are neighbors in the ring.

3 The Simulator

We now introduce HiSim. We briefly present the deployed PeerSim API followed by an overview of problems occurring, when simulating large-scale P2P networks. We provide solutions to these problems and explain the main design of the simulator. HiSim is designed for HiSbase where we use it to conduct query simulations, create large-scale networks, evaluate new protocols and gather statistics on network load distribution.

3.1 PeerSim

PeerSim is a P2P simulation environment, implemented in Java [MJ09]. Its simulation engine provides the basic functionality for simulating and managing network connections and passing messages. A network is represented as a list of nodes, each of them maintaining a list of protocol objects. Additionally, initializer objects (executed before the simulation) and control objects (passive simulation monitors executed periodically) exist. PeerSim supports two main simulation approaches, cycle-driven and event-driven. In the cyclic model, protocols are executed periodically while in the event-driven case, protocol execution is triggered by messages which are sent via the simulated transport layer. The simulation is controlled by a configuration file which offers an easy way to set the different

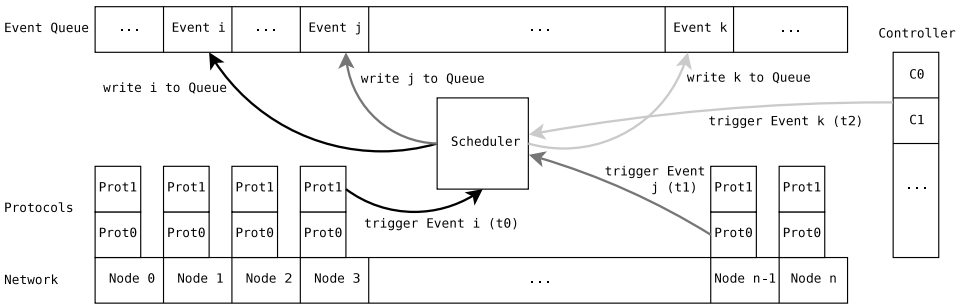


Figure 2: A sample execution of a PeerSim simulation.

simulation parameters. The PeerSim engine itself is single-threaded and all events (periodic and non-periodic) are written to one central event queue. The execution order of the events is determined by a scheduler which schedules the events according to their internal simulator time t (see Figure 2; isochromatic arrows indicate that the corresponding actions are performed in one atomic step). We decided to use PeerSim as its scalability outnumbers other common used simulators [NLB+07] and its API is well-designed and provides a good basis for extensibility. Additionally it is in wide use by the research community (see [JMJV] for a list of publications).

3.2 Design Issues for Large-Scale P2P Networks

When designing a P2P simulator architecture, usually two basic design issues arise: the choice between cycle- and event-based simulations and between single-threaded and completely parallel (one thread per node) execution.

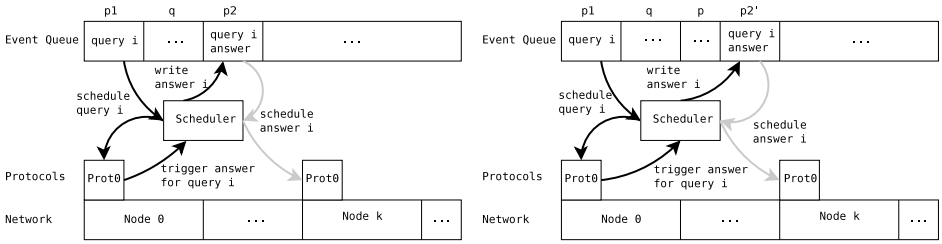
Using a cycle- or an event-driven simulation approach heavily depends on the application. A cyclic engine can be used to simulate gossiping mechanisms, swarm intelligence techniques or to gather simulation statistics periodically. Event-driven scenarios involve the simulation of messages or queries. PeerSim provides engines for both cases and even the possibility to combine them. We use the event-based engine for simulating queries and apply the cyclic model to periodically gather statistics on the network load (see section 4.2). We deploy the combined approach in one protocol which processes queries and performs periodic checks on a node's load (see section 4.1).

The choice between single- or multi-threaded takes several aspects into account. The single-threaded approach (as implemented in PeerSim) has certain advantages compared to the multi-threaded one. A sequential simulator needs no scheduling by the operating system and can be executed on standard single-core processors which limits the network size only to main memory [MJ09]. Multi-threaded simulators on the other hand need massive scheduling efforts or large high performance computing clusters to run. Otherwise, the scheduling effort grows with the networks and limits scalability. Additionally, no parallelization techniques are needed when running a single-threaded simulator which makes development and evaluation much easier. One disadvantage, however, is that sequential simulators can not achieve true realistic behavior concerning aspects like throughput measures or modeling database lookups, as the particular nodes are not autonomous. Additionally, they run on only one single processor and hence, can not utilize the full available power of common multi-core systems. Besides these two separate approaches, one could also think of a hybrid approach, combining the features of single-threaded and multi-threaded simulators. Such simulators could deploy the existing computational power by running a certain number of peers on all available processor cores in parallel. Hence, each core executes its peers sequentially, the scheduling effort stays low as the different cores are running real parallel and multi-core architectures can be used to boost the simulator scalability. This approach could also be applied to HiSim. However, developing such simulators is much more complex than developing single-threaded simulators because hybrid simulators need to be thread-safe and the particular cores have to be synchronized. Making

HiSim a hybrid simulator is subject to future work as we only present the basic simulator here. In the following section we list some problems associated with single-threaded simulations and propose solutions.

3.3 Achieve Realism in Single-Threaded Simulator Environments

In a P2P system like HiSbase a high ratio of processing time depends on the database lookup of queried data. After receiving a query, a node looks up the queried data in its database. While the other nodes keep on processing, this node waits until the lookup has been completed before sending the corresponding answer. In a parallel simulation, we can stop the thread for a certain time to easily model the lookup. In a single-threaded environment, we can not keep a node waiting as this would interrupt the whole simulation, i.e. all other nodes. To solve this problem we manipulate the PeerSim event queue. Note that in the following we omit network latencies and message delays caused by physical factors due to clarity reasons. Assume that the lookup for query i to node k is scheduled at position p_1 in the event queue. The submission of the corresponding answer is scheduled at position p_2. Then, $p_2 = p_1 + q$ with q being the number of events, scheduled between p_1 and p_2 (see Figure 3(a)). Note that these q events can be any kind of events triggered by any node or control object and have nothing to do with the query and thus, do not model a delay. They only lie between p_1 and p_2 because we operate in a single-threaded environment. In



(a) Model with no lookup delay

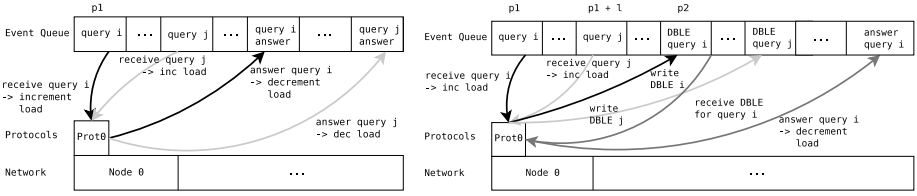
(b) Model containing a lookup delay

Figure 3: Modeling query lookup delays in single-threaded environments.

a parallel simulation, they would be processed simultaneously by different threads. Figure 3(b) now shows how we model a delay. We simply add a value d to the time, when the answer message would originally be scheduled. This postpones the message to the back of the event queue by p positions and hence $p_2' = p_1 + q + p$ holds. Consequently, the message arrives later at its target which represents our desired lookup. d can either be chosen as a constant or determined dynamically according to the amount of data the query demands.

Another problem occurs if load aspects are analyzed by the simulator. In HiSbase, each node maintains a FIFO query queue where incoming queries are listed. Currently processed queries are removed. The load of a node is defined by the number of queries, waiting in this queue. Transferring this load definition to the simulator is not possible due

to the single-threaded environment. A query comes in, is processed and the answer is submitted, all in one computational step which means that during that time no other actions can be performed by any other node and the whole simulator. Thus, no other queries can queue at the node while it processes a query. As a result, the load of a node would at most be 1. This situation is shown in Figure 4(a) (recall, that isochromatic arrows represent atomic simulation steps). To overcome this problem we could scan the simulator's



(a) Load model with load at most 1

(b) Model with correct load values

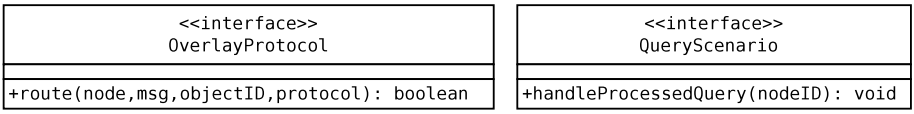
Figure 4: Modeling load in single-threaded environments.

event queue, looking for queries which have been sent to a node and have already arrived at their target. The resulting count would be the load. As the event queue can grow very large for huge networks, this is no efficient solution. Our solution is to introduce a `DBLookupEvent` (`DBLE`). Assume that a node's query request is scheduled at position p_1 in the event queue. After receiving the request the node sends a `DBLE` to itself (scheduled by the PeerSim Scheduler at p_2) but does not answer the query yet. The answer to the query is not submitted until the corresponding `DBLE` is received. The node can now queue additional query requests, scheduled at positions between p_1 and p_2 which makes load values > 1 possible (see Figure 4(b)). This load now conforms to our load definition. Note that the `DBLE` event models the time, a query waits in a node's query queue to be processed. The actual lookup delay is realized with the above described mechanism. Hence, the number of `DBLE` events scheduled for a node corresponds to the load of this node and the simulator time between p_1 and p_2 represents the time, the query waits in the queue.

3.4 Modular Architecture

Besides extending the simulation framework with aspects of query evaluation and load, our simulator is also characterized by its easy extensibility. PeerSim itself provides a solid basis for extensibility but for the HiSbase application and to keep our simulator generic, we added some new features.

For P2P systems built on top of overlay topologies, the question arises, which one to choose. Protocol behavior might differ on various overlay structures and the most suitable needs to be determined. To simplify this task, we provide the `OverlayProtocol` interface (see Figure 5(a)). This interface partially coincides with the KBR (key-based routing) API, proposed in [DZD+03]. This API comprises several operations which should be pro-



(a) The `OverlayProtocol` interface (b) The `QueryScenario` interface

Figure 5: Interfaces for overlay networks and query simulations.

vided by structured overlays, including a `route` operation. In HiSim, overlay topologies can be implemented using this interface which requires each overlay protocol to provide a `route()` method for message routing. Top-level protocols can be built on this generic overlay protocol by just deploying the generic `route()` method from the underlying overlay and hence, overlays can be exchanged simply by specifying them via the configuration file. As HiSbase is built upon Pastry, HiSim completely simulates the FreePastry[3] layer, including join and routing mechanisms.

One major part of HiSbase is its query processing functionality. Thus, the query simulation engine is an essential part of HiSim. When simulating queries, different scenarios are possible. A query scenario defines, how a node reacts after it received the answer to a previously sent query. One scenario might be that a node waits a random time period after receiving an answer and then submits a random number of queries. Another imaginable scenario is, when all nodes have batch-jobs of n queries (maximum load scenario) which they submit to the network always keeping m parallel queries in the network (*MPL* m). After receiving an answer, a node immediately submits a new query to the network to keep its m queries within the network. With our `QueryScenario` interface we provide an easy way to add new scenarios and to switch between them. Scenarios are added by implementing this interface and nodes only call `handleProcessedQuery` after they finished processing a query. The scenario then reacts accordingly. HiSim provides an implementation of a maximum load scenario and a cyclic scenario where in each cycle, a random number of queries is submitted from a random node.

4 Extensibility of HiSim

Different aspects can be thought of, when it comes to the extensibility of a given simulator. For HiSim, we show how easily new protocols can be added, statistics can be collected and additional functionalities, like storing and loading networks, can be realized.

4.1 A Sample Protocol for Dynamic Replication in HiSbase

As a first application, the simulator is used to evaluate a dynamic replication protocol (DRP) for HiSbase. Basically, the DRP works as follows. A node's current load is moni-

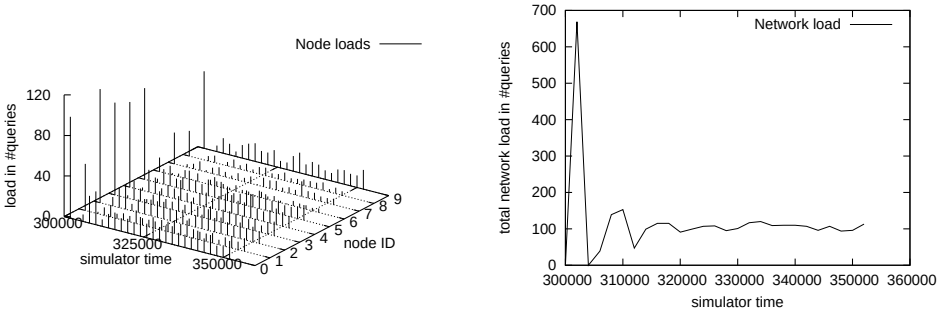
[3]http://www.freepastry.org/FreePastry/

tored periodically and if it increases above a specified level, it replicates parts of its data, placing it on other nodes determined according to their interest in the data and their load. If the load decreases again, the node revokes its replicas.

The DRP requires an extended query processing mechanism as it needs to take care of replicas. This mechanism differs from the standard processing mechanism in HiSbase and hence, we encapsulate it as one dedicated protocol. With the above described interfaces we can easily place it on top of our Pastry implementation and run it with the already implemented query scenarios without the need of further adaption. As stated above, protocols can combine the event- and cycle-driven approaches. We use this feature for the dynamic replication to perform the periodic checks on a node's load status and the query processing in the same protocol. As a result we receive a completely autonomous protocol, selectable via the configuration file and runnable instantly with different query simulation scenarios on top of Pastry. Other protocols can be added in the same way which obviously is fast and easy and keeps the design of the simulator clean. As a next task, we want to evaluate the DRP by gathering statistics from the network.

4.2 Recording Simulator Statistics

For evaluating, comparing, and analyzing different protocols, monitoring and recording statistics from the network is inevitable. We introduce a statistics component, implemented as a control object (see section 2) which periodically collects network statistics. We provide statistics on the total network load, on single node loads and on the number of queries, submitted to a HiSbase histogram region. The output format of the statistics files are tab separated values which allow convenient further processing, e.g., by using gnuplot[4]. Figures 6(a) and 6(b) show examples of such plots and were created by just passing the output files of the statistics observer to gnuplot. Adding further statistics as



(a) Statistics on node load

(b) Statistics on network load

Figure 6: Sample statistics output.

e.g. a message counter, involves only low additional effort.

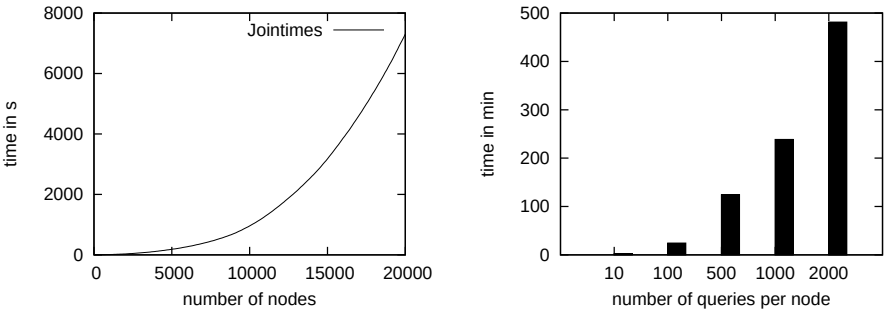
[4]http://www.gnuplot.info/

4.3 Saving and Loading Existing Network Configurations

As it is computationally intensive to create large-scale networks, our simulator provides a store and load mechanism for already created network configurations. Each simulation consists of a creation phase p_c and a simulation phase p_s. In p_s, the main simulation part, the query processing, is performed while in p_c, the network is built by using the join protocol of the specified overlay structure. When developing and evaluating new protocols it is inevitable to execute runs with the exact same characteristics as the results should be reproducible and comparable. With the store and load functionality, the state of a network (consisting of all node states) after p_c and before p_s is retained and can be reused later. Especially for large-scale networks of 10^4 or more nodes, saving a network configuration is very useful as creation time increases exponentially (see also section 5) and just reading network configurations is more than 90% faster.

5 Scalability Analysis

To analyze the scalability we performed two types of measurements. At first we measured how long network creation takes for different network sizes. We then simulated maximum load scenarios (see section 3.4) with various numbers of queries, using the previously created networks. All measures were performed on a workstation equipped with two Intel Xeon quadcore processors (2.93 Ghz) and 64 GB RAM.



(a) Jointimes for up to $2 \cdot 10^4$ nodes

(b) Querytimes for 10^4 nodes

Figure 7: Scalability results.

Figure 7(a) shows the times it takes to create a network of $2 \cdot 10^4$ nodes. Times were recorded each time 100 new nodes joined the network. We see that these times increase exponentially with the number of nodes. This is due to the fact that the more nodes exist, the more messages a node needs to send to join the network. Thus, creating a Pastry network of size 10^4 with HiSim takes about 15 minutes while $2 \cdot 10^4$ nodes already take about 2 hours. With our store-and-load functionality this effort only has to be taken once and is thus reduced to a minimum.

To simulate the query batch-jobs, we used a query set of $2 \cdot 10^4$ queries, collected from log files on a real database, storing two-dimensional astrophysical data. We conducted runs with 10, 100, 500, 1000 and 2000 queries per node. The queries were picked randomly and submitted to the network of 10^4 nodes with an MPL of 10 (for the 10 and 100 queries per node scenarios) resp. 100 (for the rest). The results in Figure 7(b) show the durations of the complete simulations. They span from 2.3 minutes for 10 queries per node to 481 minutes for 2000 queries. Looking at the results, we observe that the query simulations scale approximately linearly in the number of queries. The same setup was carried out with $2 \cdot 10^4$ nodes and the results confirm the linear scalability.

The performance measures show that we can simulate large-scale networks with high query load within a reasonable time period. The simulation durations are especially tolerable as we do not have the resources to set up such systems in reality. Additionally, evaluating different new protocols is much easier when operating on a single workstation. As a result we save a lot of time by deploying our simulator for these tasks.

6 Related Work

Many different network simulators exist to investigate P2P systems. HiSbase itself already has an integrated simulator which uses the FreePastry simulation engine. This engine allows simulating FreePastry code without any adaption but the engine is very lightweight as delays, message drops etc. are not modeled. Additionally, as the simulator is multi-threaded, network sizes of only up to 10^3 nodes are possible which brought up the need for a new simulator. Other simulator examples are NS-2 [Ber], NeuroGrid [Jos03] or GPS [YAg05] to mention only a few. NS-2 is a widely used simulator which simulates the network layer on packet-level. This is useful to analyze networks on lower layers but comes with the loss of scalability. It can be used on multiple machines and runs in parallel. GPS and NeuroGrid are two single-threaded simulation engines. NeuroGrid was initially designed for comparative simulations between Freenet-, Gnutella- and NeuroGrid-based systems. GPS is implemented in Java and completely driven by messages. No cyclic protocols are supported.

According to [NLB+07] most of the published research in P2P systems, conducted with the help of simulators, is based on custom software. This software does not deploy a standard API like the ones mentioned above or simply does not mention the underlying simulator architecture. P2PRealm [KVK+06], for example, is a simulator especially designed for studying neural network algorithms. With HiSim, we present a simulator which is built on a common basis and provides the extensibility to fit special applications.

7 Summary and Future Work

Our work was driven by the need of a P2P network simulator which can be easily extended and provides high scalability to implement and evaluate new protocols for HiSbase, a P2P

framework for handling multidimensional data. The presented HiSim meets these requirements. Its design allows adding and exchanging new protocols easily and thus, HiSim can also be deployed in other domains. It is based on PeerSim, a highly scalable P2P simulation framework which provides a single-threaded simulation engine. HiSim utilizes PeerSim to manage low level tasks like passing messages and manage physical connections. Beyond that, HiSim provides functionality especially designed for query processing applications built on top of overlay networks. The introduced design allows convenient management of various query scenarios which can easily be executed on different overlay structures. We showed that single-threaded engines are arbitrarily scalable in theory, because no thread limits caused by the operating system apply, but that other problems arise from this fact. The key idea to solve these problems is to manipulate the simulator's event queue. HiSim presents solutions to query-related problems occurring in single-threaded environments and can thus provide more realistic query behavior. With some examples we demonstrated the possibilities to extend and adapt HiSim to fit many additional tasks. We analyzed the scalability by running simulations with different node and query numbers. The results show that network creation time increases exponentially but we can reduce that effort to a minimum with the provided store-and-load mechanism. Additionally we demonstrated that the query simulation time scales linearly in the number of queries.

Future work will include more specific analyses which will compare real time system runs and simulation runs. These tests will help to determine optimal parameter settings for the query delay and for the lookup delay. We will also add new protocols and evaluate them to improve and extend the HiSbase system itself. Another major task is to equip HiSim with multi-core support.

References

[Ber] Berkeley/LNBL/ISI. The NS-2 Network Simulator. http://www.isi.edu/nsnam/ns/.

[CDHR03] M. Castro, P. Druschel, Y.C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. In *Future Directions in Distributed Computing*, pages 103–107. Springer-Verlag, 2003.

[DZD+03] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common API for structured peer-to-peer overlays. *Peer-to-Peer Systems II*, pages 33–44, 2003.

[JMJV] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. The Peersim Simulator. `http://peersim.sf.net`.

[Jos03] S. Joseph. An extendible open source P2P simulator. *P2P Journal*, 1:1–15, 2003.

[KVK+06] N. Kotilainen, M. Vapa, T. Keltanen, A. Auvinen, and J. Vuori. P2PRealm - peer-to-peer network simulator. In *2006 11th Intenational Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*, pages 93–99, 2006.

[MJ09] A. Montresor and M. Jelasity. Peersim: A scalable p2p simulator. In *Proceedings of the 9th International Conference on Peer-to-Peer (P2P'09)*, pages 99–100, 2009.

[NLB+07] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. The state of peer-to-peer simulators and simulations. *ACM SIGCOMM Computer Communication Review*, 37(2):98, 2007.

[RD01] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, volume 11, pages 329–350, 2001.

[RD10] Rodrigo Rodrigues and Peter Druschel. Peer-to-peer systems. *Commun. ACM*, 53(10):72–82, 2010.

[SBG+07] T. Scholl, B. Bauer, B. Gufler, R. Kuntschke, D. Weber, A. Reiser, and A. Kemper. HiSbase: histogram-based P2P main memory data management. In *Proceedings of the 33rd international conference on Very large data bases*, pages 1394–1397. VLDB Endowment, 2007.

[SBM+09] T. Scholl, B. Bauer, J. Müller, B. Gufler, A. Reiser, and A. Kemper. Workload-aware data partitioning in community-driven data grids. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 36–47. ACM, 2009.

[SRK09] T. Scholl, A. Reiser, and A. Kemper. Collaborative query coordination in community-driven data grids. In *Proceedings of the 18th ACM international symposium on High performance distributed computing*, pages 197–206. ACM, 2009.

[YAg05] Weishuai Yang and Nael Abu-ghazaleh. GPS: a general peer-to-peer simulator and its use for modeling BitTorrent. In *Proc. IEEE/ACM MASCOTS05*, pages 425–432, 2005.