Designing Benchmarks for P2P Systems

Max Lehn[1], Tonio Triebel[2], Christian Gross[3], Dominik Stingl[3],
Karsten Saller[4], Wolfgang Effelsberg[2], Alexandra Kovacevic[3],
and Ralf Steinmetz[3]

[1] Databases and Distributed Systems,
Technische Universität Darmstadt, Germany
mlehn@dvs.tu-darmstadt.de
[2] Praktische Informatik IV,
Universität Mannheim, Germany
{triebel,effelsberg}@pi4.informatik.uni-mannheim.de
[3] KOM – Multimedia Communications Lab,
Technische Universität Darmstadt, Germany
{gross,stingl,sandra,steinmetz}@kom.tu-darmstadt.de
[4] Real-Time Systems Lab,
Technische Universität Darmstadt, Germany
karsten.saller@kom.tu-darmstadt.de

Abstract. In this paper we discuss requirements for peer-to-peer (P2P) benchmarking, and we present two exemplary approaches to benchmarks for Distributed Hashtables (DHT) and P2P gaming overlays. We point out the characteristics of benchmarks for P2P systems, focusing on the challenges compared to conventional benchmarks. The two benchmarks for very different types of P2P systems are designed applying a common methodology. This includes the definition of the system under test (SUT) and particularly its interfaces, the workloads and metrics. A set of common P2P quality metrics helps to achieve a comprehensive selection of workloads and metrics for each scenario.

1 Introduction

In the past decade, peer-to-peer (P2P) systems have become an active research area. Originally used for file sharing applications such as Napster, P2P networks are nowadays used for various tasks like video streaming, voice communication, and gaming. It turned out that the different fields of applications require different types of P2P overlays. When new overlays are proposed, each group of researchers evaluates their system based on their individual tools and methodology, counteracting a fair comparability. Hence our goal is to develop benchmarks for P2P systems so that an unbiased comparison of different solutions can be achieved. Due to the fact that P2P networks are tailored for different purposes, the network architectures vary significantly. Thus it is not possible to create one single benchmark that is capable to evaluate every kind of P2P network. Rather it is necessary to define classes of P2P networks. Within such a class the systems

K. Sachs, I. Petrov, and P. Guerrero (Eds.): Buchmann Festschrift, LNCS 6462, pp. 209–229, 2010.

can be compared. The main challenge therefore is to understand the functionalities and interfaces of the systems that are evaluated. Based on this knowledge the classes can be defined and meaningful benchmarks can be built.

In this paper, we present exemplary benchmarking approaches for two different classes of P2P overlays. The first class are search overlays, tested using a synthetic workload on top of a minimalistic DHT interface (Section 3). DHTs are designed for large-scale applications, thus the scenario focuses on scalability (Section 4). The second class are information dissemination overlays (IDO), tested with a gaming application scenario. Unlike the DHT scenario, gaming focuses less on high scales, but rather on timing constraints which are set by a fast-paced game.

In contrast to standard benchmarks of computing systems, P2P-benchmarks have to define relevant properties of the underlay network to be reproducible. We thus propose a generic underlay model, based on commonly accepted studies on Internet node connectivity and capabilities (Section 2.4).

2 Benchmarking

The purpose of a benchmark is to quantify the performance of a system or of one of its components according to a set of quality aspects.

A useful benchmark generally should fulfill a set of basic requirements, as identified in previous work [28], [18]:

- It must be based on a workload representative of real-world applications.
- It must exercise all critical services provided by platforms.
- It must not be tuned/optimized for a specific product, i.e. it must provide a level playing field for performance comparisons.
- It must generate reproducible results.
- It must not have any inherent scalability limitations.

In the remainder of this section, we define the important terms for benchmarking and specify the quality aspects that are used for the evaluation. At the end of the section we consider issues that are specific to P2P-benchmarks and describe the underlying network model, we utilize for our benchmarks.

2.1 Terminology

System Under Test (SUT): The term System Under Test denotes the system that shall be tested or benchmarked. This system consists of several interacting components and summarizes a set of services that are offered by the service interface to a further component.

Scenario: The benchmark scenario defines the environment in which the benchmark takes place. It describes the expected functionality of the SUT and what interfaces the SUT has to provide.

Workload: The entirety of operations that are performed by the users on the SUT interface are called the *workload*. The workload specified by a benchmark can be *synthetic*, i.e., generated according to an abstracted application model. On the other hand there are *application level* workloads that are generated by real applications. A benchmark's workload is typically parametrized to be able to scale with the SUT's capabilities.

There are two common scaling dimensions in benchmarks for distributed systems (e.g., SPECjms2007 [28]), namely *horizontal* and *vertical*. Horizontal scaling affects the number of users in the system while vertical scaling affects the load generated by each user. A benchmarking system typically provides a *workload generator* component which simulates the users' behavior. The workload generator takes workload parameters as an input and operates on the SUT interface.

Metrics: To evaluate the performance of a system, it is necessary to have a set of *metrics*. Exemplary metrics are average response times, throughput, CPU consumption, or failure rates. The component that measures the system's performance according to the metrics is called *monitor*.

We differentiate between *macro metrics* and *micro metrics*. Macro metrics measure the system on its application interface level, e.g., response times of queries on a DHT. Micro metrics measure internal system aspects, e.g., routing table completeness, which help to understand *why* a system behaves a certain way. Comparing different systems using micro metrics may however be difficult due to the differences in their internal structure.

Test Procedures: In benchmarking, there are generally two basic test procedures, the *static test* and the *variation test*. The static test provides a predefined, fixed workload. Certain metrics are calculated from the system's behavior and directly account for the benchmark results. The variation test increases the load over time to push the SUT to its limits. Those limits are defined by certain QoS criteria, which are derived from system metrics (e.g., response time has to be less or equal to 2 seconds). The load is typically derived from one scalar load parameter. The benchmark result is the highest load value under which the system meets the given QoS criteria.

2.2 P2P Quality Aspects

There are certain key quality aspects that each P2P system can be evaluated for. Our benchmarks target at the quality aspects that were identified by the QuaP2P research group [3,20].

Validity, *meaning that the system responses are complete and correct.* All responses of a P2P system must match the expected responses. The term "expected" in this context means that the results obtained from the system are logically correct and complete. The system response varies depending on the type of systems being investigated. For instance, in context of search overlays the system response is the query result obtained after injecting a query request into the overlay.

Efficiency, *defined as the ratio of performance and costs.* Performance is the ability of a system to deal with its workload to a certain degree of quality. Usually this refers to the number of operations that the system can handle in a certain amount of time or the time the system needs to perform a set of operations. The term cost means how many resources are spent while the system is dealing with a given workload.

Fairness, *meaning that both the costs for operating the system and the availability of services are distributed over the participating entities (peers) such that a given fairness criteria holds.* There can be various types of fairness criteria, depending on the system type and requirements. Typical examples are uniform distribution, capacity or resource proportional distribution, and contribution proportional distribution.

Scalability, *the quantitative adaptability of the P2P system to a changing number of entities (peers or services) in the system, while preserving validity, efficiency, and fairness.* In contrast to client/server systems where resources have to be added manually in case that the system load exceeds a given threshold, P2P systems automatically scale as new joining peers share their resources with the system. Thus, scalability in the area of P2P systems is the quantitative adaptability of the system in case of a changing number of entities (peers or services), while preserving the quality aspects validity, efficiency, and fairness.

Robustness, *the persistence of a P2P system when crucial parts of a system fail.* The definition of robustness is similar to the common understanding of fault-tolerance. Robustness is, however, broader since it examines multiple failures or the failures of the participants identified to have a crucial role in the system, while fault-tolerance is a system persistence under single parts failures. Robust systems have no single points of failure, repair themselves, and failures are not propagated through the system.

Stability, *the persistence of the P2P system under system perturbations such as intensive or frequent use of system functions.* Under heavy load or load hotspots, a stable system has to maintain its functionality. In contrast to robustness, the stability definition only refers to intended system operations and does not include failures of parts of the system.

2.3 P2P Benchmarking Specifics

P2P benchmarks have certain characteristics that distinguish them from most other benchmarks. Usually, it is assumed that the SUT is self-contained, i.e., the whole system from the lower (‘physical’) hardware layers up to the software layers providing the SUT interface are part of the SUT. In contrast to that, the ‘physical world’ of P2P systems is the Internet. The Internet, however, cannot be assumed as a part of the SUT. This would contradict the requirement of reproducibility for benchmarks, since the Internet is growing every day and thus does not represent a fixed reference. While physical underlay networks can only be reproduced in small scales, it is infeasible to build a real underlay network according to the specifications provided by the benchmark. The practical approach is to simulate (or emulate) the underlay network and to run the SUT instances in the simulation. The simulator has to fulfill certain requirements (e.g., concerning simulation granularity) which must be specified in order to reproduce the simulation environment for further benchmarks, enabling comparability of results. Since it is feasible to either implement a given network specification in a simulator or to use existing simulation tools which provide the specific underlay model, the reproducibility requirement can be fulfilled. As a result, the underlying network has to be excluded from the SUT and instead specified as a part of the benchmark scenario. The following subsection describes our proposed network model.

2.4 Underlay Model Specification

In this section, we give a brief overview about current techniques to abstract and model the network with its characteristics. Subsequently, we present the chosen solution integrated in our simulation environment and detail the connection type of the hosts to the Internet within the underlay model.

Dealing with the representation of the underlay, there exist several approaches to model the underlay which may address one or several aspects of real networks like the network topology, geographical locations, delay, jitter, or packet loss. On the one hand, current models generate network topologies based on mathematical functions (e.g., Positive-Feedback-Preference-Model [34]) or topology generators like Inet-3.0 [33], where each connection between two elements exhibits its own delay and loss probability. On the other hand, some models just focus on the estimation of delays and loss probabilities of an entire communication path between two hosts, while the details of the path in between are not considered. The latter models can utilize a lookup table to estimate the delay between two communicating hosts, as proposed by Gummadi et al. [14], or following the approach of global network positioning as introduced by [24].

Regarding the utilised underlay model for our simulations, we adopt the proposed model from Kaune et al. [17] which avoids static delays and splits the calculation of the delay between two hosts up in two different parts, as depicted in Figure 1. The static part of the delay returns the minimum delay between any two hosts based on their distance to each other in the n-dimensional space. For calculating the distance between the hosts, the embedding within the space is realized using global network positioning [24]. The data for creating the positioning and for the calculation of the delay relies on the measured data of the Macroscopic Topology Project from CAIDA [1]. The dynamic part of the delay consists of the jitter for the connection between two hosts. Out of the provided data from the PingER project [4], different jitter distributions based on the geographical positions of the hosts are derived. During a simulation, a value for the jitter between two hosts is randomly chosen from the respective distribution.

Besides the realistic and accurate calculation of delay, the model of the underlying network should also take the connection type of a host to the Internet into consideration. Thus, the utilized network model for our simulations also consists of hosts with different access types, which vary regarding their available

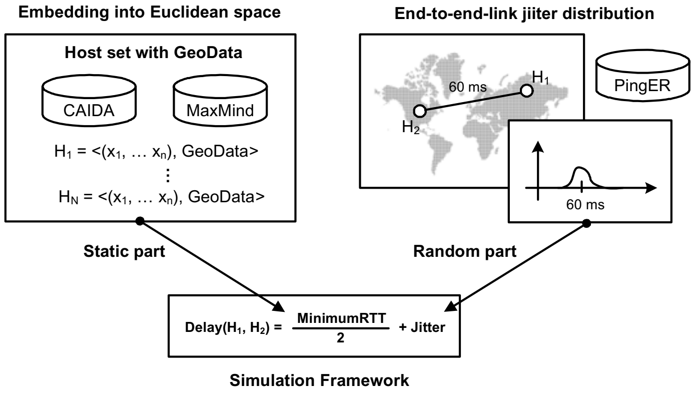
Embedding into Euclidean space
Host set with GeoData
CAIDA
MaxMind
H1 = <(x1, ... xn), GeoData>
⋮
HN = <(x1, ... xn), GeoData>
End-to-end-link jiiter distribution
60 ms
H1
H2
PingER
60 ms
Static part
Random part
Delay(H1, H2) = MinimumRTT / 2 + Jitter
Simulation Framework

Fig. 1. The calculation of delay [17]

Table 1. OECD broadband statistics from 2009

Internet connection type	Number of subscribed connections	Average upload speed (kbit/s)	Average download speed (kbit/s)
Cable modem	81.253.021	2.269	25.474
DSL[1]	168.964.115	3.055	14.404
FTTH[2]	31.589.868	51.692	76.792

bandwidth. This feature allows for simulations focusing on the heterogeneity of hosts. As an proposal for the distribution of utilized access types, Table 1 depicts an overview about the worldwide broadband access during 2009 obtained from the OECD Broadband Portal [2]. The report lists three different broadband connection types comprising the number of subscriptions and the average upload and download speed. The presented distribution of the different connections can be used as basis for a detailed model of the simulated access types specifying their characteristics and utilization. By offering the developed model, its reuse for other simulations can be facilitated and allows for obtaining comparable results.

3 Scenario 1: Distributed Hashtable

In this section we define a basic benchmark for Distributed Hashtables (DHT) which fulfills the interface presented in Section 3.1. Our benchmark addresses two main goals. Firstly, it enables the comparability of existing DHT implementations under different workloads. Based on this comparison, it is possible to determine which DHTs are suitable for a specific application scenario stating specific workload characteristics. Secondly, by pushing DHT implementations to their performance limits, their strengths and weaknesses become visible.

3.1 SUT Interface

In our scenario we focus on distributed hashtables, supporting the two basic methods `put(key, data, lifetime)` and `get(key) -> data` for storing and retrieving a data item associated with a key. The `get` function asynchronously returns the result. It is expected to always deliver a result. If no item for the requested key is found, it returns an empty data item. In most applications the key is calculated by hashing the data item using a hash function such as MD5 or SHA-1. Each data item is stored with a maximum lifetime after which it is deleted, enabling a simple garbage collection mechanism.

3.2 Workload

For the workload generation, we assume a Wikipedia-inspired document model. We model documents with a given maximum lifetime, which are stored into the DHT and retrieved afterwards. All documents that are stored in the DHT are also stored in a global document database which is not part of the SUT. Since this benchmark is designed for simulated or emulated environments, we assume that this database can be maintained as part of the global knowledge of the simulator. The database is used for selecting documents to be queried as well as for validating results obtained from the DHT. The following parameters are considered for our model:

Number of Peers, Online Time, Persistent storage on peers. An important parameter is the number of peers. The peer's online times are determined by the churn model which is described by their online time distribution. In our workload model we assume a non-persistent storage in case that a peer goes offline, which means that its stored data is deleted.

Document Size, Popularity, Lifetime. The second parameter set is related to the documents to be stored in the DHT. We model the document size and the popularity of documents based on distribution functions derived from Wikipedia article statistics. In order to avoid a constantly growing number of documents, we introduce a document lifetime after which an article is considered to be outdated and deleted from the DHT.

Peer Activity. The third set is related to the peer activity, specifying how often a peer executes a certain type of action. We define three basic operations: creating a new document, requesting an existing document, and updating a document. Hence, it is necessary to specify an execution probability per peer for each of these operations. In addition, the time between successive operations needs to be modeled based on peer activity statistics taken from Wikipedia measurements. A grace period between the creation or update of a document before a read or update request for the same document allows the DHT to properly store the documents. The delete operation is not part of the peer activity as the deletion of articles is done automatically by the DHT in case that an article's time-to-life counter has expired.

Global Document Database. We introduce a *Global Document Database* which maintains information about all documents stored in the DHT. For each document this information comprises the document id, the document lifetime, the document store timestamp, and a hash value of the document's content. This information is needed in order to verify whether the correct version of requested document is returned by the DHT. The document database offers methods for creating, updating, and requesting a document.

Per Peer Workload Generation. The workload generation algorithm which is run by each peer works as follows. Initially an activity index is defined per peer drawn from a global activity index distribution $X_{activity}$. The activity index defines the expected value of inter-arrival times between two successive actions performed by the peer. The peer action algorithm is shown in Listing 1.

```
- select action to be performed (add, modify, read)
    based on propability
- if action = add:
  - create article from global article database
  - put article to DHT
- if action = read:
  - count read action
  - get article ID to be read from global article
      database
  - get article from DHT
  - if retrieve succeeds:
    - verify document content using hash value
    - if correct: count correct response
    - else: count incorrect response
  - else:
    - count incorrect response
- if action = update:
  - perform read action
  - if read succeeds:
    - create new version of the document from the
        global document database
    - put article to DHT
```

Listing 1. Per Peer Workload Algorithm

Load Variation Schemes. In the following different load variation schemes of the workload within the DHT scenario are explained. Each scheme covers a specific situation in a DHT lifetime each assuming a different churn behavior of the peers (times without any churn, exponential churn phase, and massive join or leave).

Scheme 0: Without Churn. Peers join the network according to join function $F_{join}(t)$ until the specified number of peers n is reached. After a silent

Table 2. Workload parameters for the DHT scenario

Variable	Description	Unit	Levels
n	Number of peers in stable state (after the join process)		$[1..\infty]$
$F_{join}(t)$	Function describing the number of peers over time during the join process		$e^{\lambda t}$
t_{silent}	Duration of the graceful phase after initial join phase	s	$[0..\infty]$
X_{time}	Peer Online Time Distribution	s	Exp(λ)
$X_{activity}$	Peer Activity Distribution	s	Zipf(s)
p_{create}, p_{update}, p_{read}	Execution probability for the actions create, update, or read		$[0, 1]$, $p_{create} + p_{update} + p_{read} = 1$
$X_{popularity}$	Document Popularity Distribution		Zipf(s), Exp(λ)
X_{size}	Document Size Distribution	bytes	Zipf(s), Exp(λ)
$t_{liftime}$	Document Lifetime	s	$1..\infty$
$b_{persist}$	Whether peers persist their documents while offline		$\{0, 1\}$
$r_{massleave}$, $r_{masscrash}$	Ratio of leaving/crashing peers in case of a massive leave/crash		$[0, 1]$
$t_{massleave}$, $t_{masscrash}$	Time after the silent period after which a massive leave/crash occurs	s	$[0..\infty]$
$r_{massjoin}$	Ratio of joining peers in case of a massive join		$[0, 1]$
$t_{massjoin}$	Time after the silent period after which a massive join occurs	s	$[0..\infty]$
$t_{flashcrowd}$	Time after which the flash crowd starts	s	$[0..\infty]$
$d_{flashcrowd}$	The duration of the flash crowd	s	$[0..\infty]$
$r_{flashcrowd}$	Ratio of increased requests per peers in case of a flash crowd	s	$[0..\infty]$
$F_{request}(t)$	Function describing the variation of the inter-arrival time of per-peer actions over time.		$m * t + a$

period of t_{silent} seconds where no further join or leave of peers occurs the workload is deployed on the system. (Figure 2a)

Scheme 1: Exponential Churn. After a join phase as in Scheme 0, there is a silent period of t_{silent} seconds after which the system should be in stable state. Then, an exponential churn phase with exponentially distributed session times of the peers together with the workload is deployed on the system. (Figure 2b)

Scheme 2: Massive Leave/Crash. The third scheme covers the extreme situation of a massive leave or crash of peers. As in Scheme 1, peers join, and

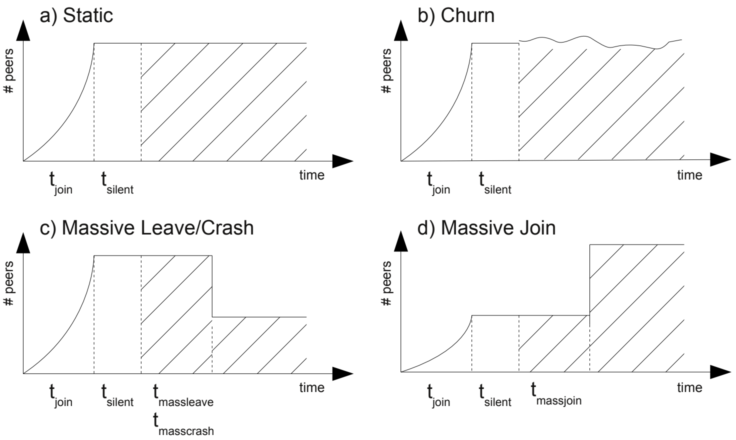

Fig. 2. DHT scenario schemes

the workload starts after a silent period. Then, after $t_{massleave}$ ($t_{masscrash}$) seconds, a massive leave (crash) with $r_{massleave}$ ($r_{masscrash}$) of leaving (crashing) peers occurs. (Figure 2c)

Scheme 3: Massive Join. Scheme 3 is similar to Scheme 2 but with joining instead of leaving or crashing peers. (Figure 2d)

Scheme 4: Flash Crowd Behavior. In this scheme, a large amount of the peers requests a specific content in a short amount of time. Based on Scheme 1, after $t_{flashcrowd}$ seconds, the flash crowd phase begins. The request frequency per peer is increased by a factor of $r_{flashcrowd}$ with a duration of $d_{flashcrowd}$ seconds.

Scheme 5: Linearly Increasing Number of Peers. Peers join according to a linear join function $F_{join}(t)$, increasing their number as long as system remains stable. The workload is deployed instantly at the beginning of the benchmark.

Scheme 6: Linearly Increasing Number of Requests. Based on Sscheme 1, the workload is deployed on the peers with a decreasing inter-arrival time as defined by the request increasing function $F_{request}(t)$.

3.3 Metrics

This section combines the load variation schemes with appropriate metrics (Table 3) to provide benchmarks for the P2P quality aspects as described in Section 2.2.

Robustness. For testing robustness, we use Scheme 2 with massively crashing peers. After $r_{masscrash}$ of the peers have crashed, the system either becomes stable again after t_r seconds or it remains unstable where the remaining peers are unable to reorganize the DHT topology. t_r is the time the system takes to return to a certain minimum acceptable QoS level. In case of our

Table 3. Metrics for the DHT scenario

Variable	Metric	Unit	Description
t_q	Average query response time	s	The time that passes between the insertion of query into the system and its response averaged over time.
q	Number of queries		The total number of all executed queries within the measurement interval.
r_+	Number of correct responses		The total number of correct responses within the measurement interval. A response is considered to be correct if and only if the right document in the latest version is returned.
r_-	Number of negative responses		The total number of incorrect responses within the measurement interval. Both invalid responses and missing responses are counted as incorrect.
S	Success ratio		The ratio of successfully executed query requests. $S = \frac{r_+}{q}$.
$\bar{S}$	Fail ratio		The ratio of failed query requests. $\bar{S} = \frac{r_-}{q}$.
t_r	Recovery time	s	The time needed by the quality metric of a system to return back to a defined QoS level after a massive perturbation of the system.
l_i	The average load on peer i	$\frac{bytes}{s}$	The bandwidth usage on peer i within the measurement interval averaged over time.
L_i	The average relative load on peer i		The bandwidth usage on peer i in relative to its maximum capacity within the measurement interval averaged over time.
μ_l	The average load on the overall system	$\frac{bytes}{s}$	The average bandwidth usage of all peers. $\mu_l = \frac{1}{n} \sum l_i$
μ_L	The average relative load on the overall system		The average relative bandwidth usage of all peers. $\mu_L = \frac{1}{n} \sum L_i$

DHT scenario the QoS level is defined by the query time t_q and the success ratio S. For instance, the average query time should always be below 2 seconds and the query success ratio should be above a threshold of 0.95 ($t_q < 2 \wedge S > 0.95$).

Efficiency. The efficiency of the system is measured using Scheme 1 where peers join and leave the system according to an exponential churn model. Furthermore, a typical workload is applied on the DHT. Efficiency is defined as the quotient of performance and costs. In case of a DHT, the performance is the average query response time t_q. Costs are defined as the average load μ_l on the system for solving the query requests. In order to calculate how efficient the system is the average query time is divided by the average system load.

Validity. In order to benchmark the validity of the results returned by the DHT we again consider a typical churn influenced environment as described in Scheme 1 and measure the success ratio S.

Fairness. Benchmarking fairness is done by applying Scheme 1 and measuring the average load l_i on each peer. For the purpose of simplification in our DHT scenario the distribution of load is considered to be fair if the load is distributed equally over all peers. More sophisticated definitions of fairness can be taken into account, e.g., a capacity-proportional definition of fairness where load has to be distributed according to the capacities of the peers. To calculate the degree of fairness the standard deviation of the relative load over all peers is calculated as $\sigma_L = \frac{1}{n-1} \sum (L_i - \mu_L)^2$.

Stability. To test the stability of a DHT, we use Scheme 1 with an increasing exponential churn factor. The stability is measured by the maximum churn level under which the query reponse times t_q and success ratio S fulfill the required QoS levels.

Scalability. The scalability of a DHT is tested by increasing the workload on the system vertically (number of request) or horizontally (number of peers) according to Schemes 5 and 6 respectively. In both cases we measure the maximum scale up to which the query response times t_q and success ratio S fulfill the required QoS levels.

4 Scenario 2: Massively Multiplayer Online Game

The area of P2P Massively Multiplayer Online Games (P2P MMOG) is much more complex and less standardized than the field of DHTs. Several groups have been doing research in P2P MMOGs in the last ten years, focusing on various aspects. Those can be categorized to six main issues [13]:

Interest Management. In an MMOG, every player has his own personal view on the game world, depending on his current state, most importantly his location. That view defines what parts of the world he can see and what events he can perceive. Interest management (IM) decides which information is necessary for each player to build his personal view of the world. The *area* *of interest* (AOI), typically centered at the player's position and bounded by his *vision range* (VR), defines the region within which the player needs to receive game event information.

Game Event Dissemination. The game event dissemination system has to ensure that each player receives all relevant game events within his AOI. Real-time games require low latencies in the event dissemination to keep the players' views as fresh as possible. Since the AOI is bound to game world positions, the dissemination systems are typically based on game world proximity. The task can thus also be formulated as a spatial publish/subscribe model.

NPC Host Allocation. Many games have the concept of so-called *non-player characters* (NPC) which are game entities controlled by scripts and/or artificial intelligence and which interact with the human players in the game. Since there is no central server the program controlling an NPC has to run on some peer. The assignment of NPC routines and states to peers and, if necessary, their relocation to alternative peers is part of the NPC Host Allocation.

Game State Persistence. Any object in the game world that is not directly associated to a player has to be kept persistent and consistent. The object state must be replicated to one or more peers in the network, and leaving peers must transfer their objects to others. An important requirement specific to games is that operations on game objects must not induce high latencies since the game cannot be paused until the operation is complete.

Cheating Mitigation. P2P games require special mechanisms for cheating prevention and reaction. There is no central server with a full view on the whole game world, thus the cheating mitigation algorithms must work in a decentralized manner without access to the complete game state.

Incentive Mechanisms. A P2P system lives from the resources provided by the participants. Those resources include network bandwidth, CPU cycles, and storage capacity. Incentive mechanisms make sure that every participant has to provide a certain amount of resources and prohibit free-riders.

In our scenario we concentrate on the first two aspects, Interest Management and Game Event Dissemination. While NPC Host Allocation and particularly Game State Persistence are topics for a future benchmark, the performance of cheating mitigation mechanisms is hardly quantifiable. Incentive mechanisms are a general topic on P2P systems, and P2P MMOG do not make special demands on these mechanisms. Therefore, they can be analyzed in separate scenarios.

4.1 SUT Interface

A typical MMOG information dissemination overlay (IDO) integrates the two issues Interest Management and Game Event Dissemination. As introduced above, Interest Management in an MMOG is based on an AOI defined by the vision range. The two systems VON [15] and pSense [29] act as references for our scenario. Both VON and pSense interpret the vision range as a (more or less constant) radius and the AOI as a circle on the 2D plane of the game world.

The overlay network topology is constructed locally by each player (i.e., each peer) using the AOI radius and the relative positions of surrounding players in the game world. Thus, an important aspect of MMOG IDOs is their awareness of player positions in the game world.

It is the purpose of the IDO to disseminate the game events generated by each player to all other players in whose AOI he currently is. Those events include first and foremost the player's movements (or, more generally, his position), but also other game-specific activities such as firing a missile. Since the IDO is aware of game world positions but should still be generic, it is necessary to split the disseminated information into position updates, which have semantics known by the IDO, and game-specific data that is opaque to the IDO. Besides the dissemination to the whole set of interested nodes, it should be possible to send messages to single nodes.

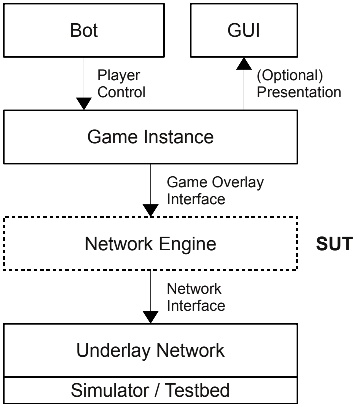

Fig. 3. The abstract game architecture

The core of our architecture (Fig. 3) consists of the *game instance* containing the local game logic and the *network engine* managing the network communication. From the benchmarking point of view, the network engine implements the SUT and the game instance applies the workload.

Based on the given requirements, our API connecting the local game logic and the network component comprises the following concepts:

- The network engine regularly pulls the local player's state (particularly its current game world coordinates) from the game instance and disseminates them. Depending on the particular IDO, the position information is also used to build the overlay topology. The pull and dissemination frequency is chosen by the network engine so that it can adapt the generated update traffic when necessary (e.g., in case of congestion).
- Neighboring players within the AOI and their positions are pushed by the network engine to the game instance whenever new information is available.

- Game actions other than player movements are pushed by the game instance to the network engine at a time when they occur, to be disseminated or to be sent to a single player. Those messages do not have a semantic meaning to the network engine; they have to be delivered reliably and without modification.

4.2 Workload

A synthetic workload appears infeasible for a gaming scenario. Games have a relatively complex and unstandardized network functionality which makes it difficult to collect measurements to derive representative workloads. While there are plenty of studies analyzing and modeling the network traffic generated by (massively) multiplayer games (e.g., [6], [7], [30]), there are only a few trying to characterize player behavior on a game activity level, such as [31].

Network traffic models of online games, since they deal with the traffic *below* the game's network component, do not contain enough information to model the workload *on top of* that component. And player behavior models or traces from real matches cannot realistically represent the game workload on the various network implementations. This is because of the high degree of interactivity, which introduces feedback loops making the player's behavior depend on the network's properties (such as message delay).

For the given reasons, we propose a workload generation process that directly originates from the game mechanics specifying the player's capabilities. For the purpose of reproducible workload generation, all players are controlled by autonomous players (*bots*) which are designed just with the goal of successfully playing the game. With this approach we can model the whole degree of interactivity which, of course, is also influenced by the SUT properties.

Gameplay Scenario. Planet $\pi 4$ [32], the game designed as a workload reference, is a spaceship first-person shooter (FPS). The game scenario consists of n players. Each player is assigned to one of m teams within which they cooperate and compete with the other teams. The game world is a 3D space in which the spaceships can freely move in all directions.

Certain strategic *points of interest* (POI) are randomly scattered within a bounded region of the game world. Specifically, a POI is a base that can be captured by the teams. For each base that a team possesses, it gains game points and/or other rewards such as weapons and energy. Once captured, a team has to defend a base by keeping players from other teams out of the base's range. To capture a base it is necessary to stay within the range of the particular base with at least one player and to prevent any other teams' players to enter that range. The bases are placed on fixed solid bodies in the space, comparable to asteroids in an asteroid field. Those also serve as obstacles in the otherwise free space.

The POIs have two important aspects concerning workload generation:

- The distribution of players in the game world is influenced by the POI. Particularly, attractive POIs will generate hotspots in player density, while spaces between the POI are expected to have low densities.

- The borders of the region containing the POI are natural borders of the effective game world without the need for artificial boundaries. Although players could move far beyond the borders, there is no incentive to do so. Limiting the effective size of the game world is necessary to be able to control the average player density.

High maximum spaceship velocities together with the hotspots allow for a high workload scalability. Thereby, the overlay implementations (SUTs) can be stressed enough to find their effective limitations.

Workload Parameters. The game scenario described above provides several parameters that can be utilized to adjust the workload.

Players and teams. Each player corresponds to a peer in the network. So the number of players (n) in the game equals the number of peers. The players are divided into m (almost) equally sized teams.

POI (bases). The bases that have to be captured by the teams cause hotspots in the player density. The hotspot magnitudes can be controlled by adjusting the values (i.e. the benefit for the possessing team) of each base separately. Each base has a range in which it can be captured and a time it takes to capture it.

Gameplay region. The gameplay region is the region within which the bases are located, thus in which the gameplay happens. Together with the total number of players, its size influences the average player density. The height of the region may be set relatively small to obtain a flat, thus pseudo-2D, game world. Pseudo-2D mode is used for gaming overlays that are only designed for a 2D world.

Ships' capabilities. The game activities (moving, shooting) are heavily influenced by the corresponding capabilities of the players' spaceships. A very important factor is the maximum velocity. All position changes affect the players' AOI and thus require certain updates in the gaming overlay. The ship's maximum forward velocity limits the rate of AOI changes. (We assume that the forward velocity is the highest velocity component and thus the most important.) Additionally, the ship's inertia limit the maximum acceleration in any direction. Missile fire events have to be delivered reliably, forming a different category than position update messages. Their rate is limited by the maximum missile firing frequency.

4.3 Metrics

Metrics are used as the performance criteria for the evaluation and have a major impact on the result of the benchmark. The goal is to choose a complete and non-redundant set of metrics with a low variability [16]. Completeness can be considered as covering the relevant quality aspects *validity, efficiency, fairness, stability, robustness, and scalability* (see Section 2.2).

In the P2P gaming scenario the central services are interest management and event dissemination. Also, we are aiming to benchmark different P2P gaming overlays. Thus, a set of macro metrics must be defined. A concrete list of metrics can be derived based on the quality aspects:

Validity. The validity of an IDO is determined by the list of AOI members. Such a list is maintained by the interest management service. Thus we define the metric for validity as correctness of the AOI member list. A first approach is to use the ratio of correct entries. An improvement can be achieved by weighting wrong entries with their importance, e.g., the distance to the player's avatar, since players mainly interact with other players or objects in their vicinity. Schmieg et al. proposed to use a position quality metric [29].

Efficiency. The efficiency of a system is defined as the ratio of performance and costs. For realtime games the performance is reflected by the responsiveness. The costs are the bandwidth that is consumed to achieve the performance. In order to calculate the efficiency quotient the responsiveness must be expressed by a responsiveness index. High latencies of events are represented by a low index, low latencies correspond to a high value.

Fairness. The fairness metric depends on a given fairness criteria. In the case of pure P2P IDOs without any incentive strategies (like pSense and VON) the distribution of load can considered as fair if the load is distributed equally to all peers.

Stability. A gaming overlay is considered stable if it reacts in a valid and responsive way under stress. If either validity or responsiveness decreases to a certain amount, the system becomes unstable. Stress is a result of high player density, velocities, and interaction rates (e.g., shooting). In order to quantify the stability, QoS criteria for validity and responsiveness must be defined. The stability index is then derived from the maximum stress parameters under which the system remains stable.

Robustness. The robustness of a gaming overlay is determined by the coherence of the virtual world. Thus, the main robustness criteria is the probability of partitions when a large fraction of the peers fails.

Scalability. Like the stability, the scalability is a second level metric. It is used to measure how a systems validity, efficiency, fairness, and robustness perform with an increasing/decreasing number of participants. For each quality aspect a threshold must be defined. The scalability metric can be formulated as maximum/minimum number of participants a system can handle without exceeding a given quality threshold.

5 Related Work

There is a wide range of benchmarks in the area of computing, starting with classic CPU benchmarks such as Dhrystone or Linpack. Other popular examples are Futuremark's 3DMark [9] for 3D graphic rendering and BAPCo SYSmark [8] for business applications. Recognized database benchmarks are defined by the Transaction Processing Performance Council (TPC) [12].

Further relevant benchmarks in the area of distributed systems are provided by the Standard Performance Evaluation Corporation (SPEC) [11], for instance

SPECjms2007 [27, 10] for message-oriented middleware systems. The SPECjms 2007 benchmark describes a supermarket supply chain application scenario. It consists of company headquarters, distribution centers, supermarkets, and suppliers communicating through the message-oriented middleware. Continuing work presents performance evaluation methods for event-based systems in general [19] and particularly for publish/subscribe systems [26].

In the area of P2P there are only a few benchmarking approaches yet. General ideas for a P2P benchmarking platform and an analysis of existing tools for P2P network simulation have been presented by Kovačević et al. [22]. A concrete benchmarking scenario for structured overlays in the context of Network Virtual Environments [21] focuses on lookup and routing latencies as well as the overlay message distribution.

Carlo Nocentini et al. present a performance evaluation of implementations of the JXTA rendezvous protocol [25]. JXTA specifies a set of protocols for P2P communication of which the rendezvous protocol provides a DHT mechanism. The paper specifies the metrics lookup time, memory load, CPU load, and dropped query percentage, as well as the parameters query rate, presence of negative queries, and type of peer disconnections (gentle or abrupt). The authors omit the specification of underlay network properties; the evaluation tests are run in a local area network whose properties are not comparable with the Internet.

A comprehensive benchmark for P2P web search engines was proposed by Thomas Neumann et al. [23]. The benchmark suggests the freely available Wikipedia content as the benchmark's document corpus. The queries, are taken from Google's Zeitgeist archive. The result quality metrics are recall and precision; efficiency is measured as query response time and network resource consumption. In contrast to most others, this work includes concrete performance properties of the network and the nodes' local disk IO. However, the network property model is simplistic, assuming a fixed latency and maximum transfer rate between all nodes, and thus not taking any kind of heterogeneity into account.

P2PTester [5] is a project aiming to provide a tool for measuring large-scale P2P data management systems. The project focus is on the applicability to various kinds of systems using a modular measurement architecture. P2PTester could be a useful tool for conducting benchmarks which particularly need to measure the connectivity traffic. The presented version of P2PTester, however, only considers a real network deployment where reproducible and Internet-like network properties are hard to achieve.

6 Conclusion and Future Work

In this paper we have discussed general requirements for P2P benchmarks, and we have presented benchmarking approaches for two very different scenarios. While the DHT scenario has a relatively clear scope and well-known functionality, the gaming scenario includes several aspects that have to be identified first and analyzed separately. Despite the wide spectrum of functionalities covered by the two exemplary scenarios, we we have shown that a common methodology can be applied.

In each scenario definition we start with a description the SUT including and the general context, its functionalities, and interfaces. Based on the SUT interface, the workload is specified, including the relevant parameters that can be used to scale the workload. Metrics then have to be defined to quantify the system behavior. Those metrics are assigned to the common P2P quality aspects which provide a general-purpose categorization of the performance criteria.

Particularly for P2P benchmarks we provide an underlay model specification that reflects the important network aspects for P2P systems derived from Internet measurements. This model can be adapted for various kinds of benchmarks for P2P, or more generally, Internet-scale distributed systems.

This work is supposed to be the starting point for a larger number of Benchmarks for various types of P2P systems. The variety of P2P solutions for different purposes should become much more tangible once there is set of system classes with clearly defined functionalities and interfaces. This categorization plus the opportunity to compare the performance of alternative solutions, significantly simplifies engineering approaches for P2P applications using existing and new solutions.

It is, however, still a long way to go towards a standardization of P2P system components that are comparable trough common interfaces. With this work we have made a first step in defining a common benchmarking methodology which can be applied for any kind of P2P system.

References

1. CAIDA - Macroscopic Topology Measurements, `http://www.caida.org/projects/macroscopic`
2. OECD Broadband statistics, `http://oecd.org/sti/ict/broadband`
3. QuaP2P Project Website, `http://www.quap2p.tu-darmstadt.de`
4. The PingER Project, `http://www-iepm.slac.stanford.edu/pinger`
5. Butnaru, B., Dragan, F., Gardarin, G., Manolescu, I., Nguyen, B., Pop, R., Preda, N., Yeh, L.: P2PTester: a tool for measuring P2P platform performance. In: ICDE 2007. IEEE 23rd International Conference on Data Engineering, Istanbul, pp. 1501–1502 (2007)
6. Chambers, C., Feng, W.-C., Sahu, S., Saha, D.: Measurement-based characterization of a collection of on-line games. In: 5th ACM SIGCOMM conference on Internet Measurement. USENIX Association, New York (2005)
7. Chen, K.-T., Huang, P., Huang, C.-Y., Lei, C.-L.: Game traffic analysis: An MMORPG perspective. Computer Networks 50(16), 3002–3023 (2006)
8. BAPCo consortium. SYSmark 2007 Preview, `http://www.bapco.com/products/sysmark2007preview/`
9. Futuremark Corporation. 3DMark Vantage, `http://www.futuremark.com/benchmarks/3dmarkvantage/`
10. Standard Performance Evaluation Corporation. SPECjms (2007), `http://www.spec.org/jms2007/`
11. Standard Performance Evaluation Corporation. SPEC's Benchmarks and Published Results, `http://www.spec.org/benchmarks.html`

12. Transaction Processing Performance Council. TPC Benchmarks, http://tpc.org/information/benchmarks.asp
13. Fan, L., Trinder, P., Taylor, H.: Design Issues for Peer-to-Peer Massively Multiplayer Online Games. In: MMVE 2009 (2009)
14. Gummadi, K.P., Saroiu, S., Gribble, S.D.: King: Estimating latency between arbitrary Internet end hosts. In: 2nd ACM SIGCOMM Workshop on Internet Measurment, pp. 5–18. ACM, New York (2002)
15. Hu, S.-Y., Liao, G.-M.: VON: A Scalable Peer-to-Peer Network for Virtual Environments. IEEE Network 20(4), 22–31 (2006)
16. Jain, R.: The Art of Computer Systems Performance Analysis. John Wiley & Sons, Chichester (1991)
17. Kaune, S., Pussep, K., Leng, C., Kovacevic, A., Tyson, G., Steinmetz, R.: Modelling the internet delay space based on geographical locations. In: 17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2009), pp. 301–310 (2009)
18. Kounev, S.: Performance Engineering of Distributed Component-Based Systems - Benchmarking, Modeling and Performance Prediction. Shaker Verlag, Aachen (2005)
19. Kounev, S., Sachs, K.: Benchmarking and Performance Modeling of Event-Based Systems. It - Information Technology 51, 262–269 (2009)
20. Kovacevic, A.: Peer-To-Peer Location-Based Search: Engineering a Novel Peer-To-Peer Overlay Network. PhD thesis, Technische Universität Darmstadt (2009)
21. Kovacevic, A., Graffi, K., Kaune, S., Leng, C., Steinmetz, R.: Towards Benchmarking of Structured Peer-to-Peer Overlays for Network Virtual Environments. In: 14th IEEE International Conference on Parallel and Distributed Systems, pp. 799–804. IEEE, Los Alamitos (2008)
22. Kovacevic, A., Kaune, S., Liebau, N., Steinmetz, R., Mukherjee, P.: Benchmarking Platform for Peer-to-Peer Systems (Benchmarking Plattform für Peer-to-Peer Systeme). It - Information Technology 49(5), 312–319 (2007)
23. Neumann, T., Bender, M., Michel, S., Weikum, G.: A Reproducible Benchmark for P2P Retrieval. In: International Workshop on Performance and Evaluation of Data Management Systems. ACM, New York (2006)
24. Eugene Ng, T.S., Zhang, H.: Towards global network positioning. In: 1st ACM SIGCOMM Workshop on Internet Measurement, pp. 25–29. ACM Press, New York (2001)
25. Nocentini, C., Crescenzi, P., Lanzi, L.: Performance Evaluation of a Chord-based JXTA Implementation. In: First International Conference on Advances in P2P Systems, pp. 7–12. IEEE, Los Alamitos (2009)
26. Sachs, K., Appel, S., Kounev, S., Buchmann, A.: Benchmarking Publish/Subscribe-based Messaging Systems. In: Yoshikawa, M., Meng, X., Yumoto, T., Ma, Q., Sun, L., Watanabe, C. (eds.) Database Systems for Advanced Applications. LNCS, vol. 6193, pp. 203–214. Springer, Heidelberg (2010)
27. Sachs, K., Kounev, S., Bacon, J., Buchmann, A.: Performance evaluation of message-oriented middleware using the SPECjms2007 benchmark. Performance Evaluation 66(8), 410–434 (2009)
28. Sachs, K., Kounev, S., Carter, M., Buchmann, A.: Designing a workload scenario for benchmarking message-oriented middleware. In: SPEC Benchmark Workshop (2007)
29. Schmieg, A., Stieler, M., Jeckel, S., Kabus, P., Kemme, B., Buchmann, A.: pSense - Maintaining a Dynamic Localized Peer-to-Peer Structure for Position Based Mul-

ticast in Games. In: IEEE International Conference on Peer-to-Peer Computing (2008)

30. Svoboda, P., Karner, W., Rupp, M.: Traffic Analysis and Modeling for World of Warcraft. In: IEEE International Conference on Communications, pp. 1612–1617 (2007)
31. Tan, S.A., Lau, W., Loh, A.: Networked Game Mobility Model for First-Person-Shooter Games. In: 4th ACM SIGCOMM workshop on Network and system support for games, p. 9. ACM, New York (2005)
32. Triebel, T., Guthier, B., Süselbeck, R., Schiele, G., Effelsberg, W.: Peer-to-Peer Infrastructures for Games. In: 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV 2008, pp. 123–124 (2008)
33. Winick, J., Jamin, S.: Inet-3.0: Internet topology generator. Technical report, University of Michigan (2002)
34. Zhou, S.: Characterising and modelling the internet topology – the rich-club phenomenon and the pfp model. BT Technology Journal 24(3), 108–115 (2006)