QoS-Controlled Dynamic Replication in Peer-to-Peer Systems

Giwon On, Jens Schmitt, Ralf Steinmetz
Multimedia Communications Lab, Faculty of Electrical Engineering and Information Technology
Darmstadt University of Technology, Germany
Tel.: +49-6151-166150, Fax: +49-6151-166152
Email:{Giwon.On,Jens.Schmitt,Ralf.Steinmetz}@KOM.tu-darmstadt.de

***Abstract* — This paper presents a study of dynamic replication for peer-to-peer networks. We take an availability-centric view on quality of service (QoS) and focus on the issues of satisfying availability requirements for distributed multimedia services running on large Peer-to-Peer (P2P) systems. We especially tackle the replica placement problem where our focus is on choosing dynamically the number and location of replicas while (1) satisfying the availability QoS requirement for all individual peers and (2) taking the intermittent connectivity of peers explicitly into account. For this purpose, we model P2P systems as a dynamic stochastic graph in which the nodes go up and down depending on their assigned up probability and issue content access events with a certain level of availability requirement. Through an event-driven simulation study we compare and evaluate replication schemes which are fully distributed and adaptive and which satisfy the availability QoS requirements. Simulation results show that (1) satisfying availability QoS requires more replicas than for only increasing the hit rate, (2) the location of replicas is a more relevant factor than their number for satisfying availability QoS, and (3) even simple heuristics can achieve reasonably high availability QoS. Our proposed replication model can be used for further study on the dual availability and performance QoS for dynamically changing, large-scale P2P systems.**

I. Introduction

The rapid popularization of Internet-based P2P applications such as Napster[1], Gnutella[2], FastTrack[3], and KaZaA[4] has inspired the research and development of technologies for P2P services and systems. While much of the attention has been focused on the issues of providing scalability, copyright solutions or routing mechanisms within P2P networks, the availability issue has so far seldom been mentioned, and there is no work known to us which tries to satisfy and guarantee the availability requirements for all individual peers.

In this paper, we present a study of dynamic replication where our goal is choosing dynamically the number and location of replicas to satisfy the availability QoS requirement for all individual peers, while taking intermittent connectivity of peers explicitly into account. In particular, the main focus of our work is building a model and devising mechanisms to study the problem of how to satisfy *different* availability requirements for distributed and replicated multimedia services in wide-area P2P systems, and to evaluate the achieved availability QoS.

In many existing works, it has been shown that the availability of distributed services and their data can be significantly increased by replicating them on multiple systems connected with each other, even in the face of system and network failures. Thus, we especially tackle the replica placement problem and study the effects of number and location of replicas on the reached availability QoS.

For this purpose, we use a concept called *quality of availability* (QoA) in which the availability is treated as a new controllable QoS parameter [5]. Based on the QoA concept, we model the P2P system as a dynamic stochastic graph. In this graph, all node and edge elements are parameterized, statistically independent of each other, with known availability and up probabilities. An availability requirement value is additionally assigned to each node so that the target replica placement problem is to find a replica set with which the availability requirements for all peers are satisfied.

Thus, the main focus of the paper is not on developing an additional, new algorithm for the replica placement problem, but instead on specifying the QoA-based dynamic replication model. However, we do not address the replica selection and update distribution issues in this work. These issues are handled in our previous work [6] where we also give a comprehensive survey on existing solutions for these problems.

The rest of this paper is organized as follows. In Section II, we describe the QoA metrics to be used for specifying and evaluating the quality of replication and some abstractions of P2P systems such as the P2P architecture, network topology, and peer characteristics. Section III presents the replica placement problem and details the replica placement model and algorithms that we used for our simulation study. In Section IV, we present our implementation methods including the simulation environment and in Section 5 we evaluate the results. Section 6 discusses related work. Finally, Section 7 concludes the paper.

II. Model and Problem Statement

A. The QoA - Basic Idea, Metrics and Parameters

The basic idea of the QoA concept is that availability can be defined as *a new controllable, observable QoS parameter*. Indeed, we move the focus of the objective function for the resource and performance optimization problems of the QoS field from satisfying transmission-dependent characteristics to satisfying availability requirements and to maximizing the total amount of time in which the required service functions work as expected and

their data are reachable. Given a set of different levels of availability requirements and a network topology with a finite number of possible replica locations, we are then interested in (a) how many replicas are needed, (b) where should they be placed, (c) whether their placement on the given topology satisfies the individually required QoA and (d) how they affect the overall service availability quality. We now define QoA metrics and present our methodology to study the QoA.

Availability is usually defined either as (a) the percentage of time during which the service is available or (b) the probability of service systems' reachability where each system has an independent failure probability. We use these definitions to specify our availability metrics used in both defining QoA requirements and evaluating reached QoA for networked services. Using these availability metrics - the percentage of successful service time and the failure probability of underlying systems and network connections, QoA guarantees can be specified in various forms similar to traditional network QoS [7]:

- *deterministic* - a service (or its data item) is reachable all the time with an availability guarantee, e.g. of 99.99 percent. This means for a service that the time duration where the service is unreachable should absolutely be no longer than 53 minutes for a year (1 year = 525600 minutes).
- *probabilistic (or stochastic)* - a service availability probability is guaranteed to be at least, e.g., 90 percent of the whole service access requests.

Actually, the exact form of QoA parameters can be specified both by applications and service providing systems. The QoA evaluation conditions that we use for evaluating satisfied QoA in the evaluation part of this work are as follows:

- *reachedQoA* - this indicates for each demanding node how much the availability requirement has been fulfilled by the selected placement *R*. For example, the required and satisfied availability values are 95% and 94%, respectively. Then, the *reachedQoA* is 0.99.
- *minSatQoA* - this is the minimum of the reachedQoA for all demanding nodes with the selected placement *R*.
- *avgSatQoA* - this is the average value of the reachedQoA.
- *guaranteedQoA* - this is a form of 'binary' QoA, i.e., the value is either 1 or 0. For a given node, when the *reachedQoA* is greater (or at least equal to) than the requiredQoA, then the *guaranteedQoA* is 1 else 0.

Table 1 shows the notation and definitions of these metrics.

Parameter	Notation	Definition
reached QoA(v)	$QoA_{rch}(v)$	the ratio of satisfied availability to required availability for node v, $\forall v \in V_R$ *with* $V_R = V \setminus R$
minSatQoA	QoA_{min}	$\min \{ QoA_{rch}(v) : \forall v \in V_R \}$
avgSatQoA	QoA_{avg}	$1/n(\sum QoA_{rch}(v))$, $\forall v \in V_R$ and $n = (\|V\| - \|R\|)$
guaranteed-QoA (v)	$QoA_{gua}(v)$	1, if $QoA_{rch}(v) \geq 1$, else 0
guaranteed-QoA	QoA_{gua}	the ratio of $\|V_{rch}\|$ *to* $\|V\|$, *where* V_{rch} *= set of nodes with* $QoA_{rch}(v) \geq 1$

TABLE 1: QOA METRICS

B. Abstractions of P2P Systems

As [8] classified, there are several different architectures for P2P systems: centralized, decentralized but structured, and decentralized and unstructured. We want to focus our replica placement problem on decentralized and unstructured P2P architectures in which there is neither a centralized directory nor any precise control over the network topology or content placement. In a P2P system there is a limited number of peers, say *N*. Each peer is assigned an up probability, that is, the fraction of time that the peer is up, and the QoA value that the peer requires when it accesses contents placed on other peers. The peers are independently up and the required QoA value may be different from other peers' QoA values. At any given time, a given peer may be up or down; it may be down because the peer's device is physically disconnected from the network. A peer is also not available when the peer service application is not launched. Each peer has private storage and shared storage. Only content in the shared storage can be accessed by the P2P community. Due to the fact that the peers may be heterogeneous - a powerful workstation, a personal computer, or an Internet-connected PDA, the storage capacity can be different between peers. We suppose that the content in a peer's shared storage is not lost when a peer goes down; when the peer comes back up, all of the content in its shared storage is again available for sharing. This is generally the case in P2P file-sharing systems such as Napster, Gnutella and KaZaA. In this case, we do not address the consistency issue, because we assume a read-only access.

We model dynamic networked P2P systems as dynamic stochastic graphs. We assign QoA values to every node of the graph, where the required QoA value and the supplying QoA value are decoupled for each node: the required QoA value is assigned at the graph creation time, while the supplying QoA value is calculated by checking the node's own availability probability value and its link degree (#of adjacent links). Furthermore, the nodes change their state between up and down according to given probability distribution functions.

The scope of dynamics that we capture in this work are peers' state (up/down) which causes the change of the number of total peers being up, their connectivity and their available storage capacity. Concerning a peer's state and availability of contents located on the peer, we can assume that the contents on the nodes are unavailable, when the peer goes down. In our P2P model which is used in the simulation study, we treat the up/down probability of each peer as (a) given as a prior knowledge or (b) unknown.

III. Dynamic Replica Placement Problem

A. Replication Model

Replication is a proven concept for increasing the availability of distributed systems. Replicating services and data from the origin system to multiple networked computers increases the redundancy of the target service system, and thus the availability of the service is increased. Important decisions for a replication management system are: *what to replicate?* (replica selection problem), *where to place the replica?* (replica placement problem), and *when and how to update them?* (update control problem). From these decision problems, we especially focus on the replica placement problem, because the placement problem does not only depend on the users' access patterns, but also on the available, continuously changing, but limited resources of peers. In this paper, we assume a partial replication in which the individual files are replicated from their original peer location to other peers, independently of each other. Thus, there may exist different numbers and locations of replicas for each original file and the access query number, peers' storage capacity and the number of live (i.e., up) peers affect the decision of creating replicas for a given content at a given service time.

B. Notations and Problem Formulation

P2P systems that consist of (storage) nodes and network connections between them can be modelled as a graph, *G(V,E)*, where V is the set of nodes and E the set of connection links. This graph is *static* if the members and the cardinality of V and E do not change else it is *dynamic*. The graph is said to be *stochastic* when each node and link are parameterized, statistically independently of each other, with known failure or availability probabilities. The replica placement (RP) problem can be distinguished into constrained RP (CRP) and unconstrained RP (URP) problems: In the CRP, there are a set of service demanding nodes *D* and a set of service supply nodes *S*, so that $D \cup S = V$ and $D \cap S = 0$ (empty set), and the replica set R can only be built from the nodes of the set S, $R \subset S$. In the URP, every node can be either a service demanding node or a service supply node, i.e., there is no *D* and *S*, and $R \subset V$. For all of our simulation running in this paper, we model the RP problem as a *dynamic, stochastic and unconstrained* graph.

We can formulate the replica placement problem as optimization problem as follows. Consider a popular P2P system which aims to increase its data availability by pushing its content or replicating the content to other peers. The problem is to dynamically decide where content is to be placed so that some objective function is optimized under a dynamic access pattern and set of peers' resource constraints. The objective function can either minimize the total number of replicas on the whole peer systems or satisfy all individual peers' QoA requirement levels. For example, we have a stochastic graph *G (V, E)* as input and eventually a positive integer number *k* as a maximum number of replicas for each content. The objective of this problem is to place the *k* replicas on the nodes of *V*, i.e., find R with $|R| = k$ such that a given optimization condition *O(|R|, R, QoA_condition)* is satisfied for given availability requirements of service demanding nodes. How well the optimization condition is satisfied depends on the size of |R| and the topological placement *R*. Because the main goal associated with placing replicas on a given network in our work is satisfying QoA which can be required in different levels, we take the availability and failure parameters as our key optimization condition, i.e., *O(|R|, R, guaranteedQoA)*. Thus, with the use of 100% of all clients', 90%-tile, and mean clients' required availability value, the optimization condition can be denoted as *O(|R|, R, 1.0)*, *O(|R|, R, .90)*, *O(|R|, R, avgQoA)*, respectively.

C. Replica Placement Algorithms

The RP problem can be classified as NP-hard discrete location problem [9]. In literature, many similar location problems are introduced and algorithms are proposed to solve the problems in this category. The heuristics such as *Greedy*, *TransitNode*, *Vertex substitution*, etc. are applied to many location problems and have shown their efficiency [10,11]. In this work, we take some basic heuristic algorithms. Yet, different variants of these heuristics and improvement techniques can be used with light modifications to enhance the efficiency and performance of our basic heuristics:

- *Random (RA)*. By using a random generator, we pick a node *v* with uniform probability, but without considering the node's supplying availability value and up probability, and put it into the replica set. If the node already exists in the replica set, we pick a new node, until the given number reaches *k*.
- *HighestAvailabilityFirst (HA)*. For each node *v*, we calculate *v*'s actual supplying availability value by taking the availability values of all adjacent edges of the node into account. The nodes are then sorted in decreasing order of their actual availability values, and we finally put the best *k* nodes into the replica set.
- *HighUpProbability (UP)*. The basic principle of the *UP* heuristic is that nodes with the highest up probability can potentially be reached by more nodes. So we place replicas on nodes of V in descending order of up probability. The use of the UP and HA heuristics assumes that we have a priori knowledge about the network topology.
- *HA+UP*. This method is a combination of the *HA* and *UP* algorithms. For this algorithm, we first calculate the average values of up probability and supplying availability for all peers. We then select those nodes as replica nodes for which both values are greater than the average values: we first check the availability probability value and then the up probability value.
- *Local*. To replace or create a new replica during service runtime (simulation runtime), the peer places a new replica on its local storage.

IV. Simulation

A. Simulation Environment

We built an experimental environment to perform a simulation study for the replica placement problem addressed in Section III. Our goal in conducting an availability evaluation is to study the effect of changing $|R|$ and R on the required and reached QoA which is given as the optimization condition, for example $O(|R|, R, avgSatQoA)$. For our availability evaluation, we conducted simulations on random network topologies. By using the LEDA library [12] several random topologies in different sizes can be generated at run time.

The availability and failure probability parameters for nodes of the graphs are one dimensional values: for example, 50, 80, 90 or 99% as availability values and 10, 5, or 1% as failure probability values. We decoupled the availability values between the demanding and supply nodes, i.e., all nodes have two availability parameters assigned: one value as the demanding availability parameter and the other as the supplying. Thus, when a node is a demanding node (issuing query events to access content), then its demanding availability value is used, while for a supplying node the actual supplying availability value is, for example, calculated by multiplying the availability values of its own and the average value of its adjacent edges. At the replica set building phase, each node is evaluated according to its supplying availability value. Thus, to be elected as a replica node, for example in the *UP* algorithm, a node should have a high up probability value.

The simulation program is written in C/C++ and runs on Linux (Suse 8.0) and Sun Solaris 2.6 machines.

Parameter	Values
test graphs	*G1(100,300), G2(1K,5K)*
peer up probability	*0.0 - 0.9 (avg: 0.3)*
peer's storage capacity	*100, 500, 1000 MB*
content (data file) size	*3, 5, 10, 100 MB*
content popularity	*.01 - .99*
QoA required	*.50 - 0.99*
QoA supplying	*.51 - 0.99*
number of peers	*100, 1000*
number of origin contents	*1000*
number of query events	*1000*
query distribution	*Uniform*
number of simulation time slots	*100*
initial placement	*Random, UP, HA+UP*
replica replacement	*Local, UP, HA+UP*

Table 2: Simulation parameters and their value ranges

V. Evaluation and Discussion

In this section we present our experiment results. We evaluated the reached QoA of the used schemes using topologies of different sizes as well as parameter values shown in Table 2. We ran each simulation on each topology using different value ranges for the availability and up probability parameters of nodes. The demanding and initial supplying availability values of the nodes, as well as the up probability values of the nodes are assigned randomly, from a uniform distribution. To evaluate the QoA offered by our replication schemes, we used the QoA metrics defined in Table 1 of Section II.

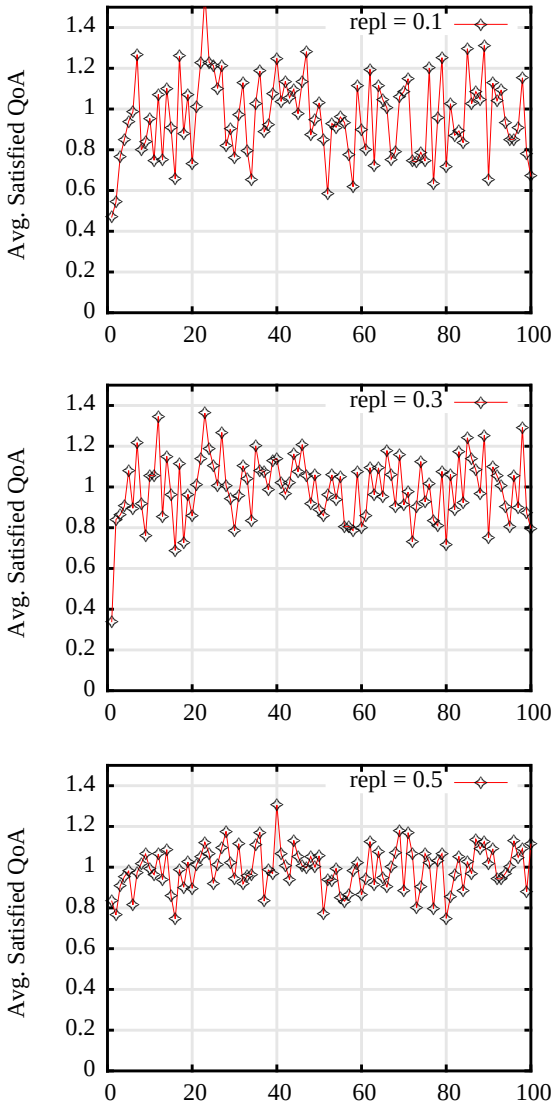

Figure 1: *avgSatQoA* with *Random* placement: #peers=1000, peers' up probability=0.3 and *Local* replacement policy. x-axis means simulation time slot.

A. *Effects of |R| on Reached QoA*

The first experiment examines how the number of replicas affects the reached QoA. For this purpose we fixed the peers' average up probability as 0.3. The simulation starts by placing *k* distinct contents randomly into the graph without considering peers' up probability. Then the query event generator starts to generate events according to a Uniform process with average generating rate at 10 queries per simulation time slot. For each query event, a peer is randomly chosen to start the query. As search method, we use a multi-path search algorithm which finds all redundant paths from the querying peer to all peers that have the target content (inclusive replica). The replacement policy for Figure 1 is *Local* replacement, i.e., in the case when the current storage capacity is not enough to store the new replica, the querying peer deletes old replicas from its shared storage. Figure 1 shows the results from this experiment with the test graph G2. We plot the simulation time slot on the x-axis and the reached QoA (*avgSatQoA*) on the y-axis. We distributed the 1,000 query events randomly on the 100 simulation time slots. As Figure 1 shows, by increasing the replication ratio, the average satisfied QoA values are converging towards 1. This means, on the other side, that the number of peers which contain the requested content (or its replica) on their own local storage is proportional to the replication ratio.

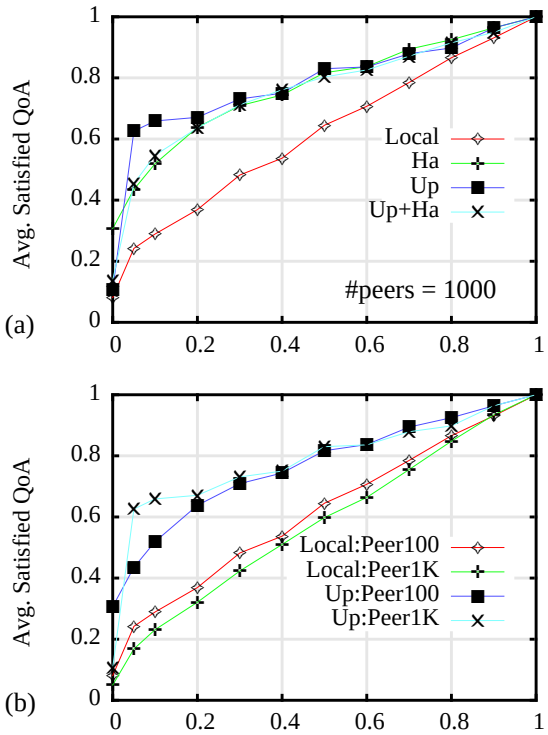

Figure 2: Satisfied QoA with Random placement:
peers' up probability=0.3.
x-axis means replication ratio, 0-100%

B. *Effects of Replacement Schemes on Reached QoA*

In the second experiment we took different replacement schemes that create new replicas during the simulation run when the reached QoA with existing replicas from the up peers at the given time slot does not satisfy the required QoA. In addition to the *Local* replacement policy, we tested the three heuristics *UP*, *HA*, and *UP+HA* with the assumption that we have knowledge about the peers' state. As Figure 2 shows, even though the heuristic algorithms are very simple, they achieved considerably higher AvgSatQoA than the *Local* scheme. For example, the QoA improvement of the replication ratio range 10-50 is about 30-70%. Figure 2 (b) shows that this improvement pattern is observable independent of the graph size: Peer100 and Peer1K in Figure (b) are equal to the nodes size 100 (graph G1) and 100 (graph G2), respectively.

C. *Satisfied QoA versus Hit Probability*

Maximizing hit probability is one frequently used goal for content replication [13]. In Figure 3 we show a comparison between the two goals, i.e., satisfying required QoA and maximizing hit probability. In this comparison the hit probability is increased when the querying peer finds the target content, while for satisfying QoA the peer should additionally check the reached QoA by calculating all the reachable paths to the peers containing the target content (or replica). We run the simulation on the test graphs G1 (P100) and G2 (P1K). The average up probability of peers is fixed again as 0.3 and we used *Random* and *UP* placement schemes for initial and replacement phase, respectively. As Figure 3 shows satisfying required QoA incurs higher cost, i.e., more number of replicas than just maximizing hit probability. For example, at replica rate=0.2, the gap between Avg*SatQoA* and *Found* (hit probability reached) is about 20% of achieved rate. And, to achieve the same rate of 80%, for satisfying QoA, we need a 30% higher replication ratio.

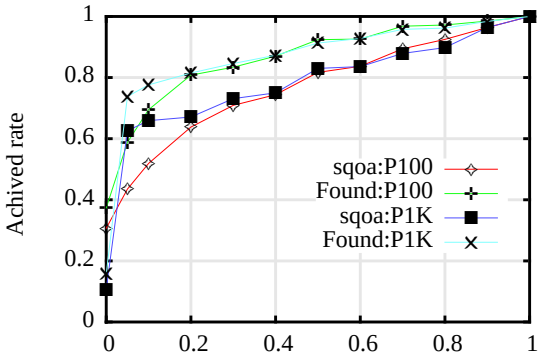

Figure 3: Satisfied QoA with *Random* placement
and *Local* replacement scheme:
peers' up probability=0.3.
x-axis means replication ratio.

The following observations could be identified from our experimental results: (1) the location of replicas is a relevant factor for satisfying the QoA. While the QoA improvement could be achieved by increasing replica numbers, replica location and their dependability affected the QoA more significantly; (2) Even a simple heuristic-based dynamic replica (re-)placement could increase the reached QoA.

VI. Related Work

Replication related works that have recently been published are [8,13,14] where the goals are somewhat different; maximizing hit probability of access requests for the contents in P2P community, minimizing content searching (look-up) time, minimizing the number of hops visited to find the requested content, minimizing replication cost, distributing peer (server) load, etc. Kangasharju et al. [13] studied the problem of optimally replicating objects in P2P communities. The goal of their work is to replicate content in order to maximize hit probability. They especially tackled the replica replacement problem where they proposed LRU (least recently used) and MFU (most frequently used) based local placement schemes to dynamically replicate new contents in a P2P community. As we have shown in Figure 3, maximizing hit probability does not satisfy the required QoA and, furthermore the two different goals lead to different results. Lv et al. [8] and Cohen and Shenker[14] have recently addressed replication strategies in unstructured P2P networks. The goal of their work is to replicate in order to reduce random search times.

Yu and Vahdat [15] have recently addressed the costs and limits of replication for availability. The goal of their work is to solve the minimal replication cost problem for a given target availability requirements, thus they tried to find optimal availability for given constraint on replication cost where the replication cost was defined to be the sum of the cost of replica creation, replica tear down and replica usage. Our work differs in that our goal is to replicate content in order to satisfy different levels of QoA values required by individual users. Furthermore, their work does not take P2P system specific features such as changing peers' state - going up or down - into account.

Related to supporting lookup services, there are many ongoing research efforts such as Chord [16]. They detail the mechanisms for supporting the services that they offer such as indexing, lookup, insert, search, update, and delete. While some of them support fault tolerance by replicating the mapping information, i.e., the key/value binding information on multiple peers, they do not give any availability guarantee for values, e.g., files or multimedia contents, than that of 'best-effort' availability support. Furthermore, it is not clear under which criterion the number and location of replicas are determined.

VII. Conclusion

In this paper we presented our modelling and simulation studies of dynamic replication strategies for decentralized P2P systems. We took an availability-centric view on QoS and treated availability as a new controllable QoS parameter. Based on the QoA concept, we modelled a P2P system as a dynamic stochastic graph where all nodes are parameterized with known availability and up probabilities. We tackled the replica placement problem and studied the effects of the number and location of replicas on the reached QoA. Our goal was choosing dynamically the number and location of replicas to satisfy the availability QoS requirement for all individual peers, while taking intermittent connectivity of peers explicitly into account. From simulation studies, we have learned that (1) satisfying QoA requires more replicas than only increasing hit rate, (2) the location of replica is a more relevant factor than its number for satisfying the required QoA, and (3) even simple heuristics can achieve reasonably high QoA. For a practical use of our proposed model, we can adopt a service and resource monitor located in each peer, which gathers periodically the necessary availability-related information such as total service launch time and percentage of freely available storage space, etc.

VIII. References

[1] Napster. http://www.napster.com/.

[2] Gnutella. http://www.gnutella.com/.

[3] FastTrack. http://www.fasttrack.nu/.

[4] KaZaA. http://www.kazaa.com/.

[5] G. On, J. Schmitt and R. Steinmetz. "The Quality of Availability: Tackling the Replica Placement Problem for Multimedia Service and Content." in *LNCS 2515 (IDMS-PROMS'02)*, pp. 313-326, Coimbra, Portugal, Nov. 2002.

[6] G. On, J. Schmitt and R. Steinmetz. "Design and Implementation of a QoS-aware Replication Mechanism for a Distributed Multimedia System," in *LNCS 2158* (IDMS 2001), pp.38-49, Sep. 2001.

[7] J. Schmitt. *Heterogeneous Network QoS Systems*. Kluwer Academic Publishers, June 2001. ISBN 0-793-7410-X.

[8] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. "Search and replication in unstructured peer-to-peer networks." In *Proc. of the 16th annual ACM International Conf. on Supercomputing (ICS'02)*, New York, USA, June 2002.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[10] N. Mladenovic, M. Labbe and P. Hansen: "Solving the p-Center Problem with Tabu Search and Variable Neighbourhood Search", <http://www.crt.umontreal.ca/>

[11] S. Jamin, C. Jin, A. R. Kurc, D. Raz, Y. Shavitt. "Constrained Mirror Placement on the Internet", In *Proc. of IEEE INFOCOM'01*, pp. 31-40, 2001.

[12] LEDA - the library of efficient data types and algorithms. Algorithmic Solutions Software GmbH. software available at <http://www.algorithmic-solutions.com/>

[13] J. Kangasharju, K.W. Ross, and D.A. Turner. Optimal Content Replication in P2P Communities. Manuscript. 2002.

[14] E. Cohen and S. Shenker. Replication Strategies in unstructured peer-to-peer networks. In *Proc. of ACM SIGCOMM'02*, Pittsburgh, USA, Aug. 2002.

[15] Haifeng Yu and Amin Vahdat, "Minimal Replication Cost for Availability" In *Proc. of the 21th ACM Symposium on Principles of Distributed Computing (PODC)*, July 2002.

[16] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. "Chord: A scalable peer-to-peer lookup service for Internet applications," In *Proc. of ACM SIGCOMM'01*, San Diego, USA, Aug. 2001.