The Effectiveness of Realistic Replication Strategies on Quality of Availability for 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
Email:{Giwon.On,Jens.Schmitt,Ralf.Steinmetz}@KOM.tu-darmstadt.de

Abstract

In this paper, we take an availability-centric view on quality of service (QoS) and propose a model and mechanisms for studying the effectiveness of realistic replication schemes on availability QoS for peer-to-peer (P2P) systems. We especially tackle the dynamic replica placement (RP) problem where our focus is on choosing dynamically the number and location of replicas while (1) meeting different availability QoS requirements for all individual peers and (2) taking the intermittent connectivity of peers explicitly into account. We model P2P systems as a dynamic stochastic graph in which the nodes go up and down depending on their assigned up probability. We develop some simple heuristic algorithms for solving the RP problem, which are fully distributed and adaptive. Through an event-driven simulation study we compare and evaluate the achieved availability QoS of the proposed RP algorithms. Simulation results show that (1) even simple heuristics can achieve reasonably high availability QoS, and (2) satisfying availability QoS requires more replicas than for only increasing the hit rate.

1. Introduction

Recently, it has been realized that the importance of satisfying service availability is becoming one of the most critical factors for the success of Internet-based services and applications [1]. In this paper, we present a study of dynamic replication for availability, 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. 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. Some selected characteristics of P2P systems, which motivate this paper are:

- Peers go up/down independently on each other. They are connected to a P2P network for a while and become disconnected after doing some service-related operations, e.g., downloading contents.
- Peers are symmetric in terms of supplying and demanding services or content. This means that there is no peer which is permanently serving other peers, and vice versa.
- Peers demand and supply *different levels* of service availability. The fact, whether a peers has launched the P2P system's program and whether the peer has still enough storage capacity or access link bandwidth, affect strongly the supplying availability of the peer.
- The availability level, that peers demand at service access time, differs between peers; some peers may expect extremely high available access, while other peers may be happy with '*best-effort*' availability level.

We model the P2P system as a dynamic stochastic graph. In this graph, the nodes go up and down depending on their assigned uptime probability and issue content access events with a certain level of availability requirements. We refine the traditional availability definitions which are limited to reasonably quantify achieved availability of (wide-area) P2P systems. This is because the traditional definitions are mostly used to specify the service uptime of tightly-coupled or clustered distributed systems. Thus they are neither suited to explicitly capture the supplying availability of individual system components nor to cover failures of communication links between peers.

The placement algorithms considered in this paper are simple heuristics which use a 'ranking'-based approximation method. I.e., compute the achieved availability of placing one extra replica on one peer node for all peers and sort these achieved availability values and select the best one that does not violate any constraint of the used metrics. To quantitatively study the effectiveness of the proposed placement algorithms, we develop an event-driven simulation model which captures the data access model as well as peers' dynamic behaviour, e.g., going up or down, etc.

Through the simulation study, we learn that even simple heuristics can achieve reasonably high availability QoS, and that satisfying availability QoS requires more replicas than only increasing hit rate. Additionally, the simulation results indicate that the location of replicas is a relevant factor for satisfying the availability QoS. While the availability QoS improvement could be achieved by increasing replica numbers, replica location and their dependability affected the availability QoS more significantly. Our proposed replication and simulation model can be used for further study on the dual availability and performance QoS for dynamically changing, large-scale P2P systems, as well as on the replica placement for availability QoS guarantees.

The rest of this paper is organized as follows. In Section 2, we describe the proposed refinements of availability and the availability QoS metrics to be used for specifying and evaluating the quality of replication. Section 3 presents the replica placement problem and details our target P2P system model, the replica placement model and algorithms that we use for our simulation study. In Section 4, we present our implementation methods including the simulation environment and results. Section 5 discusses related work and Section 6 concludes the paper.

2. Availability

2.1. Traditional Definitions

Availability is one of the most important issues in distributed systems. Traditional definitions of availability are typically based on (a) how reliable the underlying system is, (b)whether the system has any built-in features of failure detection and recovery, and (c) whether the system has any redundancy for its individual system components ([4]). In traditional distributed systems, service availability is usually defined as (a) the percentage of time during which the service is available (Equation 1).

$$Availability = \frac{MTTF}{MTTF + MTTR} \quad \text{with} \qquad (1)$$

```
failure: no P2P service
MTTF: mean time to failure
MTTR: mean time to repair
```

However, these traditional availability definitions cannot explicitly capture the availability of individual system components or the reachability of any data required by the system, in particular when all these individual system components which affect the quality of supplying service availability have different failure levels. For example, an availability value of 99% does not indicate, whether it is due to the failures of any disks or system nodes. Furthermore, since these definitions are mostly used to specify the availability values for tightly-coupled or clustered distributed systems, especially when they are applied to widely distributed (P2P) systems, they do not cover failures of communication links between peers. As a consequence, we need to refine the traditional availability definitions to capture the availability of all the individual system components. In Section 2.2 we propose three availability refinements, *fine-grained*, *decoupled* and *differentiated* availability.

2.2. Refining Availability Definition

While we keep the traditional availability definitions as a basis for our availability study, we refine them to enable to specify all the individual availability requirement levels between different users, as well as to quantitatively evaluate the reached availability of widely distributed systems.

Fine-Grained Availability

We refine the traditional availability definition as follows:

$$Avail_{Service} = Avail_{Data} \times Avail_{System} \quad \text{with} \quad (2)$$

$$Avail_{System} = Avail_{Node} \times Avail_{Link} \quad \text{and} \quad (3)$$

$$Avail_{Node} = Avail_{dynamic} \times Avail_{intrinsics} \quad (4)$$

This fine-grained availability definition contains the following meanings:

- a service is available when both its data and the system on which the service is running are available.
- a data is available when it is reachable at access time.
- a system is available, when both, nodes and communication links, are available.
- a link is available, when it does not fail and there is enough resources which can be allocated for transmitting the requested data stream for the demanding application.
- a node is available, when it is up, i.e., not disconnected from the network, and its intrinsics can be allocated for processing the service request. Memories, CPU cycle, and storage spaces are examples for such kind of intrinsics.

Decoupled Availability: Demanding versus Supplying

We separate availability levels which the service (or the underlying system) supplies from the availability levels which users (or applications) request and perceive. Thus, when we have an availability level of five nines (99.99%) we can declare whether it is a requested availability value by the users or a supplied value by the service system. By having this refinement one can check whether the service system maximizes availability, as well as whether the service system satisfies the requested availability.

For specifying demanding availability, we re-use the availability definition where availability is defined as a ratio of successful accesses to totally requested accesses. For example, a demanding service availability of 99.99% means that a user expect to have an availability level of at least, 99.99 percent of the whole successful service access requests. The demanding availability levels can be specified directly by users at service access time or by means of Service Level Agreements (SLAs) which may be a service contract between users and service providers. In comparison to the demanding availability, the supplying service availability can be calculated by using Equation (2)-(4).

Differentiated Availability

In P2P systems where several multiple applications are hosted, the availability levels required by different applications may usually vary. I.e., not all applications require the highest availability level of 'five nines', but instead an appropriate level which satisfies the application specific requirements. A similar phenomenon can be observed within a single application in which individual users demand different level of availability quality due to any resource or cost limitations. Here, we summarize some selected motivations for differentiating availability levels:

- Different users require different availability levels.
- Different services and contents have different importance priority levels.
- Availability levels are affected by different times of day.

Availability Metrics

To compare and evaluate the achieved availability among the proposed replication strategies in this work, we use the *quality of availability* (QoA) concept [5] where the availability is defined as *a new controllable, observable* quality of service (QoS) parameter. 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 achieved QoA in the evaluation part of this work are as follows:

Table 1: QoA Metrics

Parameter	Notation	Definition
satisfied QoA(v)	$QoA_{sat}(v)$	the ratio of supplied availability to demand availability for node v, $\forall v \in V_R$ *with* $V_R = V \backslash R$
minSatQoA	QoA_{min}	$\min\{QoA_{sat}(v) : \forall v \in V_R\}$
avgSatQoA	QoA_{avg}	$1/n(\sum QoA_{sat}(v))$, $\forall v \in V_R$ and $n = (\lvert V\rvert - \lvert R\rvert)$

- *satisfiedQoA* - this indicates for each demanding peer how much the availability requirement has been fulfilled by the selected placement *R*. For example, the required and supplied availability values are 95% and 94%, respectively. Then, the *satisfiedQoA* is 0.99.
- *minSatQoA* - this is the minimum of the satisfiedQoA for all demanding nodes with the placement *R*.
- *avgSatQoA* - this is the average value of the satisfiedQoA.

Table 1 shows the notation and definitions of these availability metrics.

3. Dynamic Replica Placement in P2P Systems

3.1. P2P System Model

Basic Assumptions

As the architecture for our target P2P system of this work, we basically assume a decentralized and unstructured architecture in which there is neither a centralized directory nor any precise control over the network topology or content placement. At this point we assume that the P2P system runs over an overlay network where each peer's physical connection link can be mapped to a logical link in the overlay network. Furthermore, each peer, like a single Autonomous System and BGP router of the Internet, has the ability to manage multiple routing paths to any destination peer to access service concents, either the original or replicas. Thus, when the destination peer or any peer among the path crashes or the (sub)path goes down, it can see other operational paths and choose the best one to continue its service access.

Modelling P2P Service Systems as Stochastic Graphs

P2P systems that consist of peer nodes and interconnection links between them can be modelled as an *undirected graph*, *G(V,E)*, where V is the set of nodes and E the set of connection links. This graph is *dynamic* if the members and the cardinality of V and E change else it is *static*. The graph is said to be *stochastic* when each node and link are parameterized, statistically independently of each other, with known availability or failure probabilities. For all of our simulation running in this paper, we model the target P2P system as a *dynamic and stochastic* graph.

In this graph, we assign the availability values to every node of the graph, where the demanding and the supplying availability value are decoupled for each node: the demanding availability value is assigned at the graph creation time, while the supplying availability value is calculated by Equation (4). Furthermore, the nodes change their state between up and down according to the given probability distribution function.

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 the 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, we treat the up/down probability of each peer as (a) given as a prior knowledge or (b) unknown.

3.2. Replication Model

In this paper we assume a partial replication model in which the individual files are replicated from their original peer location to other peers, independently of each other. Important decisions for a replication system, which we will intensively study in this work are:

- *what to replicate?* - replica selection. Selecting target replicas depends on the popularity and importance of contents, which can be gained by tracing users' access histories. To build a realistic access model, the *Uniform* and *Zipf*-like query distributions [8,14] are adopted for our simulation study. As content access type we assume a read-only access. This is generally the case in P2P file-sharing systems such as Gnutella [2] and KaZaA [3]. In this case, we *do not address the consistency issue*.
- *how many to replicate?* - replica number. In addition to the popularity and importance of contents, the storage capacity and access bandwidth of peers affect strongly the decision of the number of replicas. In this work, we also capture the number of replicas under replication, i.e., the number of peers that have a particular content. We use the term, replication ratio to mean the percentage of nodes having the content. For example, replication ration 0.1 means that 10% of all peers have a replica of the original content. To fix the number of replicas during the initial placement phase of our simulation runs, we will use the static replica distributions, *Uniform* and *Proportional*, as given in [8].
- *where to place the replicas?* - replica location. As [5] shows, the location of replicas is a more relevant factor than the number of replicas for achieving high QoA. Furthermore, to find a 'good' placement we should take not only contents' popularity or peers' storage/link capacity into account, but also the availability of individual peers, e.g., the number of live (i.e., up) peers which may have the original content or its replicas to be accessed. Our replica placement model consists of two phases, *proactive* and *on-demand* placement. The proactive placement is done at service initialization time before any content access query is issued, while the on-demand placement occurs during service run time. We model the proactive placement to be performed with/without *a prior* knowledge about the content popularity and the network topology. In case of the on-demand placement, new replicas are created, if the set of currently reachable replicas (including the original content, if available) does not satisfy the demanding availability value of the querying peer. Additionally, some existing replicas may be replaced by the new replicas, if there is a storage capacity problem at peers on which the created replicas should be placed.

3.3. Problem Formulation

We formulate the replica placement problem as optimization problem as follows. Consider a P2P system which aims to increase its service 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 the dynamics of content access pattern and peers' availability and 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., $O(|R|, R, satisfiedQoA)$ *or* $O(|R|, R, avgSatQoA)$.

3.4. 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*.
- *HighlyUpFirst (UP)*. The basic principle of the *UP* heuristic is that nodes with the highest uptime probability

can potentially be reached by more nodes. So we place replicas on nodes of V in decreasing order of uptime probability.

- *HighlyAvailableFirst (HA)*. For each node *v*, we calculate its actual supplying availability value by taking all the availability values of its data, intrinsics and of all its adjacent edges 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. The use of the UP and HA heuristics assumes that we have a prior knowledge about the network topology.
- *Combined (HA+UP)*. This method is a combination of the *HA* and *UP* algorithms. For this algorithm, we first calculate the average values of uptime 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 uptime probability value and then the availability probability value.
- *Local*. To create or replace a new replica during service runtime (i.e., simulation runtime), the peer places a new replica on its local storage. The replica replacement policy bases either on *least recently used* (LRU) or on *most frequently used* (MFU) concept.

4. Simulation

4.1. Simulation Methodology

We built an experimental environment to perform an event-driven simulation study for the replica placement problem addressed in Section III. 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. Table 2 summarizes the simulation parameter settings and the random number functions used for our simulation. The simulation program is written in C/C++ and runs on Linux (Suse 8.0) and Sun Solaris 2.6 machines.

Table 2: Simulation parameters and their value ranges

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
range of demand availability values	.50 - 0.99

Table 2: Simulation parameters and their value ranges

Parameter	Values
range of supplying data availability	.51 - 0.99
number of peers	100, 1000
number of origin contents	1000
number of query events	1000
query distribution	Uniform, Zipf
number of simulation time slots	100
proactive placement heuristics	Random, UP, HA, HA+UP
on-demand placement heuristics	Local-LRU, UP, HA, HA+UP

4.2. Simulation Results and Evaluation

We evaluated the satisfied 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 parameters of nodes. The demanding and initial data 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 2.

Effects of |R| on Satisfied QoA

The first experiment examines how the number of replicas affects the satisfied 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 the *Uniform* process with average generating rate at 10 queries per simulation time slot. For each query event, a peer is randomly chosen to issue 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 (either the original or a replica).

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 average satisfied 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, the number of peers which contain the requested content (or its replica) on

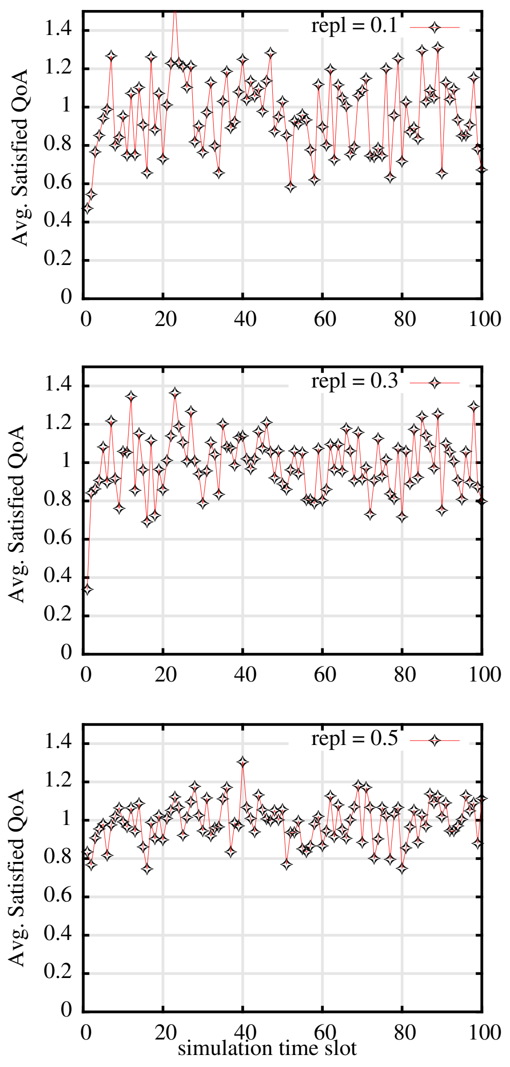


Figure 1: Effects of replication ratio on satisfied QoA where proactive placement: *Random*, #peers=1000, peers' up probability=0.3, and on-demand placement: *Local-LRU*. X-axis means simulation time slot.

their own local storage is proportional to the replication ratio.

Effects of Initial Replica Selection on Satisfied QoA

In the second experiment we compared the two replica selection schemes - *Uniform* and *Proportional* which decide, for a given fixed number of *k*, the target replicas among original contents at the service initialization phase. In this experiment we placed the *k* replicas on randomly chosen peers which do not contain the original content of the corresponding replica. Furthermore, the peer contains only one replica for each original content. As Figure 2 shows, the *Proportional* scheme offers higher satisfied QoA than the *Uniform* scheme for the *Zipf*-like access query model.

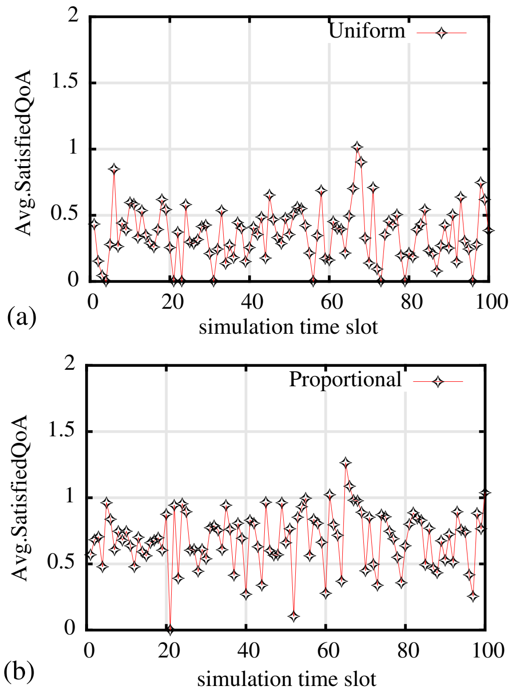


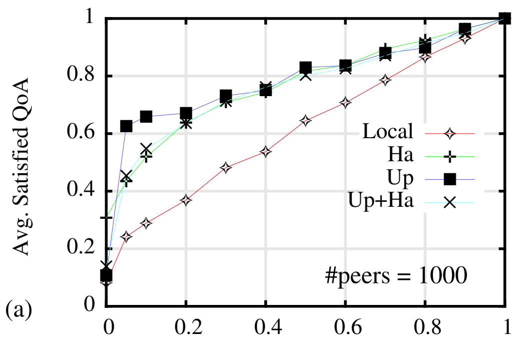
Figure 2: Effects of initial replica selection schemes on satisfied QoA: proactive placement: *Random*, #peers=1000, peers' up probability=0.3, and query model: *Zipf*.

Effects of Placement Schemes on Satisfied QoA

In the third experiment we took different on-demand schemes that create new replicas during the simulation run when the supplied QoA with existing replicas from the up peers at the given time slot does not satisfy the demanding QoA. In addition to the *Local* scheme, we tested the three heuristics *UP, HA,* and *UP+HA* with the assumption that we have knowledge about the peers' state. As Figure 3 shows, even though the heuristic algorithms are very simple, they achieved considerably higher satisfied QoA than the *Local* scheme. For example, the QoA improvement of the replication ratio range 10-50 is about 30-70%. Figure 3 (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 1000 (graph G2), respectively.

Satisfied QoA versus Hit Probability

Maximizing hit probability is one frequently used goal for content replication [13]. In Figure 4 we show a comparison between the two replication goals, i.e., satisfying re-



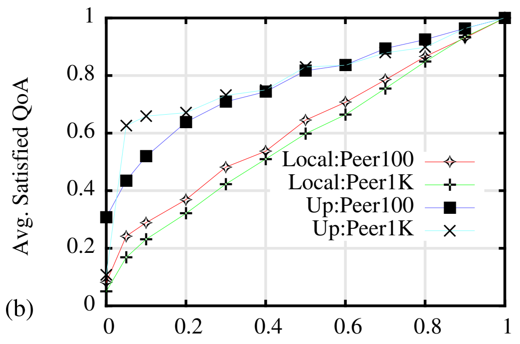


Figure 3: Effect of placement strategies on satisfied QoA where proactive placement: *Random* and peers' up probability=0.3. (a) average satisfied QoA from all four heuristics used. #peers=1000, (b) a comparison of the average satisfied QoA between *Local-LRU* and *UP* heuristic with different graph sizes. The number of peers of Peer100 and Peer1K is100 and 1000, respectively. X-axis means replication ratio, 0-100%.

quired 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 supplied 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 and G2. The average up probability of peers is fixed again as 0.3 and we used *Random* and *UP* placement schemes for proactive and on-demand phase, respectively.

As Figure 4 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 *sqoa* (satisfied QoA) 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.

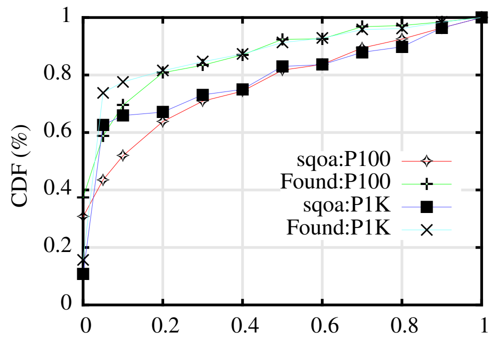


Figure 4: Comparison of replication cost for different replication goals: satisfying QoA vs. maximizing hit probability. P100 and P1K mean 100 and 1000 nodes, respectively. X-axis means replication ratio. Y-axis means the comulative distribution function of the achieved QoA (sqoa) and hit rate (Found).

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 satisfied QoA.

5. 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 4, 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] and Pastry [17]. 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.

6. Conclusion

In this paper we presented our modelling and simulation studies of dynamic replication strategies for satisfying availability in decentralized P2P systems. We took an availability-centric view on QoS and treated availability as a new controllable QoS parameter. We modelled a P2P system as a dynamic stochastic graph where all nodes are parameterized with known availability and up probabilities. Based on the QoA concept, 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. Our proposed replication and simulation model can be used for further study on the dual availability and performance QoS for dynamically changing, large-scale P2P systems, as well as on the replica placement for availability QoS guarantees.

Furthermore, 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.

7. References

[1] H. Schulzrinne. "QoS over 20 Years". Invited Talk in *IWQoS'01*. Karlsruhe, Germany, 2001.

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

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

[4] G. Coulouris, J. Dollimore and T. Kindberg. *Distributed Systems*, 3rd Ed., Addison-Wesley, 2001.

[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, 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 Pub., 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. 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] 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.

[17] A. Rowstron and P. Druschel. "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," In *Proc. of the IFIP/ACM International Conf. on Distributed Systems Platforms,* Oct. 2001.