[OSSt03] *Giwon On, Jens Schmitt, Ralf Steinmetz;* **Quality of Availability: Replica Placement for Widely Distributed Systems;** Eleventh International Workshop on Quality of Service IWQoS 2003, Montrey, CA, USA, Juni 2003, S. 325-342.

Quality of Availability: Replica Placement for Widely Distributed Systems

Giwon On, Jens Schmitt, Ralf Steinmetz

Multimedia Communications Lab (KOM), Department 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
http://www.kom.e-technik.tu-darmstadt.de/

Abstract. In this paper, we take an availability-centric view on Quality of Service (QoS) and focus on the issues of providing availability guarantees for widely distributed systems such as web servers and peer-to-peer (P2P) file sharing systems. We propose a concept called *Quality of Availability* (QoA) in which the availability is treated as a new controllable QoS parameter. The newly refined fine-grained availability definitions and QoA metrics enable the specification and evaluation of the different level of availability for different users and applications. We tackle specifically the replica placement (RP) problem where our focus is on choosing the number and location of replicas while (1) meeting *different* availability QoS requirement levels for all individual users and (2) taking the intermittent connectivity of system nodes explicitly into account. We decompose the RP problem into two sub-problems: (1) *improving* QoA and (2) *guaranteeing* QoA. We investigate a number of simulations - for *full* and *partial* replication models and *static* and *dynamic* placements - to compare and evaluate the achieved availability QoS of the developed RP algorithms. Our proposed QoA concept and model can be used as a base mechanism for further study on the effectiveness of realistic replication schemes on both availability and performance QoS for widely distributed systems.

1 Introduction

Even though there are many significant research results, technology advances and solutions in *Quality of Service* (QoS) in the last 20 years [1,2], their application to commercial products or systems has not been so successful in comparison with their attention in the research arena. One probable critical reason is that, as pointed out in [3], the main research focus for QoS was to control transmission characteristics like bandwidth, delay, and loss. This is because Internet applications which typically assumed the need for QoS support, such as video-on-demand (VoD) and Internet telephony, strongly motivated the development of QoS technologies. While for these the control of the transmission characteristics is certainly important it seems likely by now that, on the one hand, for them this may not be the most pressing need with regard to QoS requirements, and on the other hand that there are other applications having quite different requirements. Indeed, the perceived QoS may be much more influenced by how available a certain service and its data are. In the context of QoS, availability as an issue has so far seldom been mentioned, and there is no work known to us which tries to treat availability as a controllable QoS parameter.

Concerning (service) availability support, while most research efforts in high availability and fault-tolerant systems areas focus on achieving the so-called 'five nines' (99.999%) availability [4], there is a demand for service differentiation from service consumers and providers due to costs and competitive nature of the market space, which derives for the mechanisms that support different levels of services and their availability.

The work in this paper is strongly motivated by the two aspects mentioned above - importance of *service availability* and *differentiation* of service classes and availability requirements. As a consequence, we take an availability-centric view on QoS and focus on the issues of providing availability guarantees in widely distributed systems and services. For this purpose, we propose a concept called *quality of availability* (QoA) where the availability is treated as a new controllable QoS parameter. Based on the QoA concept, service providers and consumers can specify and check the target levels of service availability that they offer and require, respectively, in a fine-grained form. To enable these features, we first refine the traditional availability definitions which are limited to reasonably quantify achieved availability of widely distributed 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.

We then tackle the replica placement (RP) problem where the main goal is to choose the number and location of replicas to satisfy (and eventually guarantee if required) *different* level of availability QoS requirement for all individual users while taking the intermittent connectivity of system nodes explicitly into account. We decompose the RP problem into two sub-problems: (1) *improving* QoA and (2) *guaranteeing* QoA. For improve QoA, we take simple ranking-based heuristics which calculate the supplying availability for all service hosting nodes and select the nodes with higher (or highest) availability values as replica nodes on which the replicas are placed. To guarantee QoA, we develop an exact method *state enumeration* which enumerates all possible placements without skipping any solution case. The algorithm can actually guarantee QoA but has exponential run-time complexity. Thus, we develop an additional algorithm called *admission-controlled placement* which also offers a QoA guarantee but has significantly low run-time complexity.

To quantitatively study the effectiveness of the proposed placement algorithms, we run and analyse a number of simulations - for *full* and *partial* replication models and *static* and *dynamic* placements. For the dynamic placement we develop an event-driven simulation model which captures the data access model as well as systems' dynamic behaviour. Simulation results show that (1) even simple heuristics can achieve reasonably high availability QoS, but they cannot give any guarantee for their achieved availability QoS, (2) the state enumeration algorithm guarantees the availability QoS with its placement results, but the run-time complexity is exponential, and (3) satisfying availability QoS requires more replicas than only increasing the performance, e.g., increasing hit rate.

The rest of this paper is organized as follows. In Section 2, we describe the proposed refinements of availability definitions, the QoA concept and its metrics which are used for specifying and evaluating the quality of replication. Section 3 presents the replica placement problem. We details our target system and replica placement model and the proposed algorithms. In Section 4, we present the simulation study and results. Section 5 discusses related work and Section 6 concludes the paper.

2 Availability Refinement and QoA Metrics

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 ([5]). In traditional distributed systems, service availability is usually defined as the percentage of time during which the service is available (Equation 1).

$$Availability = \frac{MTTF}{MTTF + MTTR} \quad \text{where} \tag{1}$$

```
MTBF = MTTF + MTTR
failure means no distributed service
MTBF is the mean time between failure
MTTF is the mean time to failure
MTTR is the 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 systems, they do not cover failures of communication links between system nodes. 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 the specification of all the individual availability requirement levels between different users, as well as to quantitatively evaluate the reached availability of widely distributed systems.

2.2.1 Fine-Grained Availability

We refine the traditional availability definition as follows:

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

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

$$Avail_{Node} = Avail_{NodeDynamics} \times Avail_{NodeIntrinsics} \tag{4}$$

This fine-grained availability definition captures the following:

- 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 are enough resources which can be allocated for transmitting the requested data 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. Resources such as memory, CPU cycle, and storage spaces are examples of such intrinsics.

2.2.2 Decoupled Availability: Demand versus Supply

We distinguish between availability levels which the service (or the underlying system) supplies from the availability levels which users (or applications) request and perceive. This refinement enables one to check whether the service system maximizes availability, as well as whether the service system satisfies the requested availability.

For specifying demand availability, we re-use the availability definition where availability is defined as a ratio of successful accesses to totally requested accesses. For example, demand service availability of 99.99% means that a user expects to have an availability level of at least 99.99 % of the whole successful service access requests. The demand 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 demand availability, the supply service availability can be calculated by using Equation (2)-(4).

2.2.3 Differentiated Availability

In widely distributed 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 levels of availability due to resource or cost limitations. We now 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 the time of day.

Figure 1 shows the three refined availability definitions proposed above.

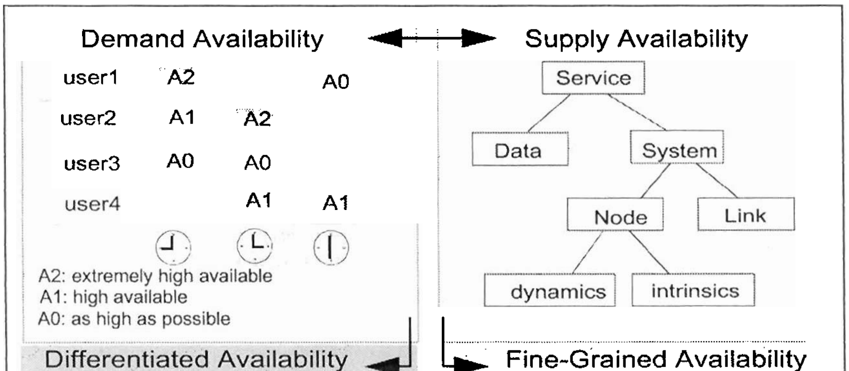

Figure 1: Illustration of the refined availability definition. The left side, ‘demand availability’ can be easily mapped to a single service usage scenario in which the different availability levels (A0 - A2) are affected by the time of day (9 am, 3 pm and 6 pm).

2.3 The Concept of Quality of Availability (QoA)

2.3.1 Basic Idea and Goals

The basic idea of the QoA concept is that the 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 such as minimizing transmission delay, jitter, and/or loss to satisfying the availability requirements such as minimizing failure time of service systems and their components and to maximizing the total time in which the required service functions as expected and its data are reachable.

The goal of our work is to understand and satisfy quality of availability (QoA), i.e. to maximize service systems' requested service time and to control and guarantee QoA. 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 how many replicas are needed, where should they be placed, whether their placement on the given topology satisfies the individually required availability QoS and how they affect the overall service availability quality. In the following section we define QoA metrics.

2.3.2 QoA Metrics

To compare and evaluate the achieved availability among the proposed replication strategies in this work, we define and use the following QoA metrics (see Table 1):

- *satisfiedQoA* - this indicates for each demanding peer how much the availability requirement has been fulfilled by the selected placement *R*. For example, if the required and supplied availability values are 95% and 94%, respectively, the *satisfiedQoA* is 0.99.
- *guaranteedQoA* - it indicates for how many demanding nodes the selected placement *R* satisfies the QoA requirement.

Table 1: QoA Metrics: V is set of entire system nodes of a widely distributed system and R is set of replica nodes (i.e. a placement) where $R \subseteq V$. |V| and |R| are cardinality of the node sets V and R, respectively.

Parameter	Notation	Definition	E.g.
satisfiedQoA	$QoA_{sat}(v)$	the ratio of supplied availability to demanding availability for node v	0.95, 1.05
minSatQoA	QoA_{min}	min $\{QoA_{sat}(v) : \forall v \in V \backslash R\}$	0.9
avgSatQoA	QoA_{avg}	$1/n(\sum QoA_{sat}(v)),\ \forall v \in V \backslash R$ where $n = (\|V\| - \|R\|)$	0.95
guaranteed-QoA(v)	$QoA_{gua}(v)$	for each demanding node v, availability guarantee: $A_R(v) = 1$, if $QoA_{sat}(v) \geq 1$	1 or 0
guaranteed-QoA	QoA_{gua}	the ratio of $\|V_{sat}\|$ *to* $\|V\|$, where V_{sat} = set of nodes with $QoA_{sat}(v) \geq 1$	0.9

3 Replica Placement in Widely Distributed Systems

3.1 System Model

3.1.1 Target System Features and Basic Assumptions

We take peer-to-peer (P2P) systems as the target distributed system of this work. Some selected characteristics of the P2P systems, which are considered in this paper are:

- Peers go up/down independently of each other. They are connected to a P2P network for a while and become disconnected after doing some service-related operations, e.g., downloading or uploading contents.
- Peers demand and supply *different levels* of service availability. The fact, whether a peer has launched the P2P program and whether the peer has still enough storage capacity or access link bandwidth, affect strongly the supplied availability.
- The availability level, that peers demand at service access time, differs between peers; some peers may expect extremely high available access, while others may be happy with '*best-effort*' QoA level.

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 (AS) and BGP router of the Internet, has the ability to manage multiple routing paths to any destination peer to access service contents, 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.

3.1.2 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 (|V| and |E|) 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, we model the target P2P system as a *undirected stochastic* graph where the placement can be made in both *static* and *dynamic* modes.

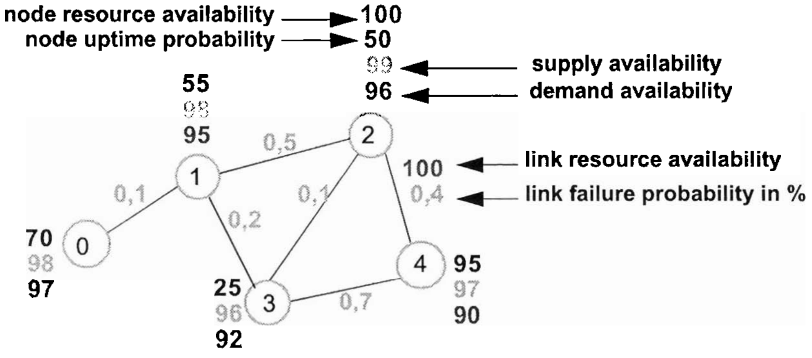

Figure 2: A distributed service modeled as a stochastic graph G(V,E), with example figures.

In this graph, we assign the availability values to every node of the graph, where the demand and the supply availability 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, in a dynamic graph, 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. Figure 2 illustrates an example stochastic graph that models a distributed system such as P2P system.

3.1.3 Replication Model

Replication is a proven concept for increasing the availability for 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. In this paper we capture both *full* and *partial* replication models. In the full replication model, the entire data of an origin (server) system is replicated to other nodes located within the same network. *Mirroring* is a typical case of the full replication model. In the partial replication model, the individual data is replicated from its original system location to other systems, independently of each other. Important decisions for these replication models, which affect strongly the achieved QoA are:

- *what to replicate?* - replica selection. Selecting target replicas depends on the popularity and importance of content, which can be gained by tracing users' access history. To build a realistic access model, the *Uniform* and *Zipf*-like query distributions [6,7] are adopted for our simulation study of the dynamic placement mode. As content access type we assume read-only access. This is generally the case in P2P file-sharing systems such as Gnutella [8] and KaZaA [9]. 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. 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 [6].
- *where to place the replicas?* - replica location. As [10] 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 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, some 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 new replicas, if there is a storage capacity problem at peers on which the created replicas should be placed.

3.2 Problem Statement

We formulate replica placement as an 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 given access model 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 such that a given target condition *O(|R|, R, QoA_condition)* is optimized for given availability requirements of the service demanding nodes. How well the target condition is optimized 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 target, i.e. *O(|R|, R, satisfiedQoA) or O(|R|, R, guaranteedQoA)* for the two RP sub-problems, *improving* QoA and *guaranteeing* QoA, respectively.

3.3 Replica Placement Algorithms

The RP problem can be classified as NP-hard discrete location problem [11]. In literature, many similar location problems are introduced and algorithms are proposed to solve the problems in this category. In this section we propose two algorithms that can be classified according to the two conditions described in Section 3.2.

3.3.1 Ranking-based Heuristics for Improving QoA

To improve QoA, we take some basic heuristic algorithms. We note however not different variants of these heuristics and improvement techniques can be used with small modifications to enhance the efficiency and performance of our basic heuristics. A short description of each of the used heuristics is as follows:

- *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 up probability can potentially be reached by more nodes. So we place replicas on nodes of V in descending order of up probability.
- *HighlyAvailableFirst (HA)*. For each node *v*, we calculate its actual supply 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.

- *HighlyAvailableFirst with HighestTransitNode (HA+TR)*. This method is a combination of the *HA* algorithm with *TransitNode* concept. The basic principle of the *TransitNode* concept is that nodes with the highest (in/out) degrees, i.e., the number of connection links to adjacent nodes, can potentially reach more nodes with smaller latency. So we place replicas on nodes of V in descending order of (in/ out) degree.
- *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.

3.3.2 Exact Method - *State Enumeration* for Guaranteeing QoA

Guaranteeing QoA is likely to satisfy a certain, required QoA value with a guarantee. This means we have to always offer a replica set which fulfils the given QoA requirements for all demanding nodes in any case. In comparison to the problem of improving QoA described above in Section 3.3.1, this problem requires a solution which exactly tests all possible placements and finds the optimum, i.e. a placement that offers a QoA guarantee with a minimal number of replica nodes. Some similar work is described in the literature, which are devoted to the problem of *network reliability* [12]. The methods that provide an exact reliability are called exact methods, in contrast to the heuristic methods which provide an approximate result. From some exact methods we adopt the *state enumeration* method [12] and modified it for our problem.

In the state enumeration method, the state of each node and each edge are enumerated: the state value is either *1* when it functions or *0* when it fails. Indeed, there are $2^{|V| + |E|}$ states for a graph $G = (V,E)$, i.e., $2^{|V| + |E|}$ partial graphs for G. We then check the QoA for all partial graphs with all instances of replica sets.

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 3. For our availability evaluation, we conducted simulations on random network topologies. By using the LEDA library [13] several random topologies in different sizes can be generated at run time. The simulation program was written in C/C++ and tested under Linux and Sun Solaris.

We ran a number of simulations - both for *full* and *partial* replication models and *static* and *dynamic* placement approaches. The dynamic placement consists of the proactive and on-demand placement phases, while the static placement considers only the proactive placement. We then compared and evaluated the achieved QoA of the developed RP algorithms using topologies of different sizes as well as parameter values shown in Tables 2 (for the static approach) and 3 (for the dynamic approach).

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 2.3.2.

Table 2: Simulation parameters and their value ranges: these values are used for the simulation runs in the static placement mode in Sections 4.2 and 4.3. The data availability for a given node is 1, if the node contains the original data or its replica.

Type	Parameter	Value
Graph	node and edge size	G1(20:30), G2(100:300)
Edge	edges' failure probability	1 ~ 10%, 0%
Node	nodes' demanding availability	90~99%, 50~99%, 50-90-99%
Data	data availability	1 or 0

4.2 Improving QoA - Static Approach

In this simulation study, we evaluate the achieved QoA by our simple heuristics. As replication model, we assumed *full replication*. We further assumed that the failure probabilities of nodes and links are known. To calculate the supplying service availability, we considered only the system's availability and assumed that the data is available when the system is available. The baseline for our experiment is an initial placement R_0 which is obtained by randomly selecting k nodes from V. We then compare the achieved QoA of each heuristic to this baseline and present the relative QoA improvement obtained with each heuristic.

4.2.1 Effects of Number (|R|) and Location (R) of Replicas on Achieved QoA

We experimented to find good locations of a replica set R with $|R| = k$ for given graphs G with maximal replica number k. The conditions that we assumed for this problem were: (1) *minSatQoA > 0.9, 0.95,* and *0.99,* respectively, and (2) *avgSatQoA > 1.0*. In this case, there was no constraints on the topological location of the replicas and replicas may be placed at any node v in G.

Figure 3 shows the results from this experiment with *G2*. We plot the number of k on the x-axis and the reached QoA on the y-axis. In each graph, we plot different curves for different heuristics and different ranges for required availability values. From Figure 3, we can see that the heuristics *HA* and *HA+TR*, although they are very simple, reach significantly higher QoA in comparison to the baseline placement. For example, at the placement with 10 replicas, our both heuristics achieved ca. 100% higher satisfied QoA in average than the Random method. On the other hand, even though the improvement of 12% QoA guarantee rate with replicas 5 to 25 (totally, 20% of the whole nodes are replicas) may not seem much, it is important to note that the number of replicas is really a relevant factor for improving QoA: the lager the replica number is, the better is the reached QoA.

4.3 Guaranteeing QoA - Static Approach

The goal of this second simulation study is to find optimal selection with a guaranteed QoA for all demanding nodes. We take the same assumptions to the simulation study of Section 4.2: *full replication* model, the failure probabilities of nodes and links known as a prior information, and the same service availability scope for supplying availability.

Avg. Satisfied QoA: demand QoA: 50-99%, link failure probability: 0-10%

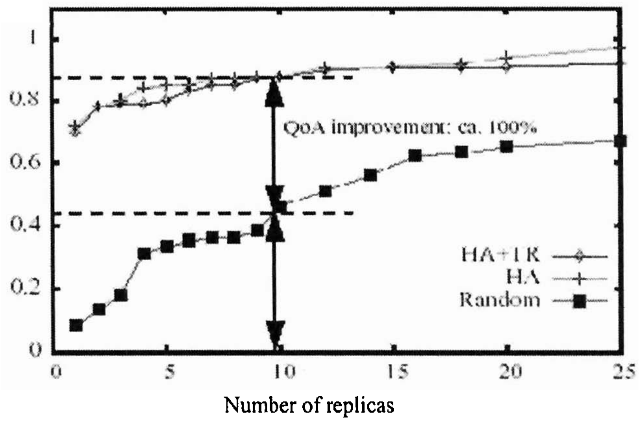

Figure 3: Achieved QoA values by our heuristics. y-axis means the satisfied QoA in average.

In this experiment, we used the *state enumeration* algorithm. Due to the exponential growing run-time complexity and the memory requirements with growing graph sizes, we limited our experiments for the *state enumeration* to a small graph, the test graph *G1* with |V| = 20 and |E| = 30. We started the routine with a replica degree of 1, i.e., *k=|R| = 1*, and selected each node as replica node. We then incremented the replica degree, until we reached the *guaranteedQoA* = 1.0 (a QoA with guarantee). Table 3 shows the achieved QoA values at each *k (k=1,2,3)*. Figure 4 plots the reached QoA that the *state enumeration* algorithm calculated exactly with each instance for the given *k*. Figure 4 shows significantly how the achieved QoA varies, and how big the gap between good and bad QoA rates reached by the instances is.

Table 3: A test result from state enumeration algorithm with G1, failure probability: 0%, and req. availability range: 90-99%. 'QoA value' means the guaranteed QoA value in average.

|R|	Best QoA value	Worst QoA value	Mean QoA value	Instances achieved the best QoA value
1	0.80	0.10	0.3345	{0},{8}
2	0.95	0.15	0.8078	{0,11},{0,18},{8,11},{8,18},{11,13},{12,16},{13,16}
3	1.00			{0,11,16},{0,16,18},{8,11,16},{8,16,18},{11,12,16},{11,13,16}

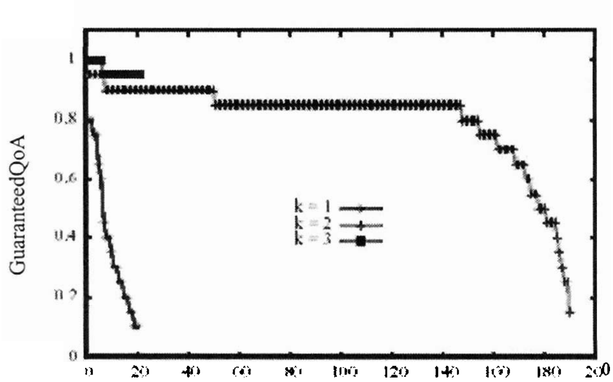

Figure 4: Achieved QoA checked exactly by the *state enumeration* algorithm. x-axis means the number of instances of R with different $k = |R|$: sorted by QoA decreasing order.

4.4 Improving QoA - Dynamic Approach

The main goal of this simulation study is to choose dynamically a 'good' placement which increase/maximize the satisfied QoA for a given replica number. We developed an event-driven simulation model which captures the data access model as well as peers' dynamic behaviour, e.g., going up or down, etc. As replication model, we assumed the *partial replication*. However, we did not assume any a prior knowledge more for the failure probabilities of nodes and links. Furthermore, we used the whole scope of the service availability definition (Equation 4) to calculate the supplying service availability. We modelled the dynamic placement in two phases: *proactive* and *on-demand*. The proactive placement is likely the static placement of Sections 4.2 and 4.3, while the on-demand placement means a replica replacement that is done by each demanding peers, i.e., content access query issuing peers, when the supplying QoA does not satisfy the demanding QoA. Thus, the dynamic placement is a decentralized and on-line placement. As placement algorithms, we used Random, HA, UP, combined HA+UP and Local-LRU. Table 4 summarizes the simulation parameters with their values used for the simulation study in this Section. 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).

4.4.1 Effects of Initial Replica Selection on Satisfied QoA

In the first 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

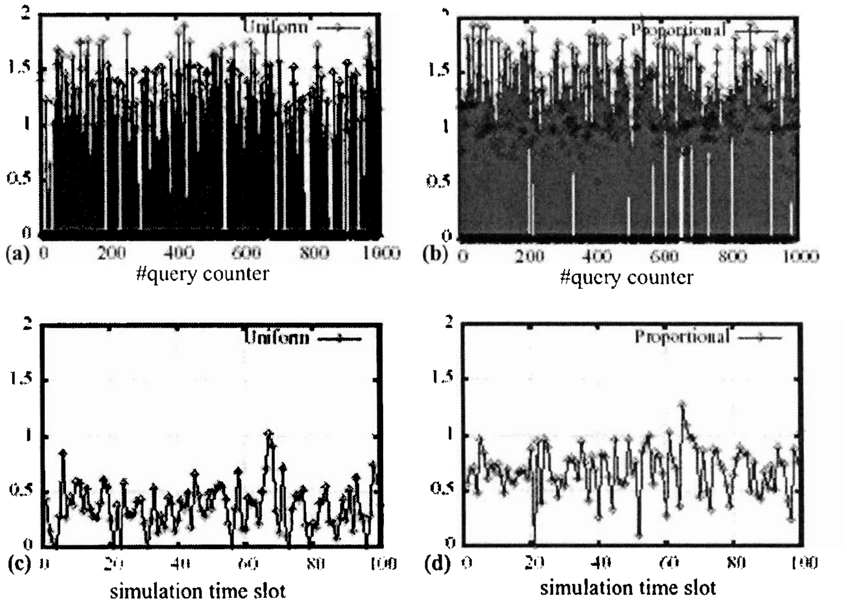

Figure 5: Effects of initial replica selection schemes on satisfied QoA with proactive placement: *Random*, #peers=1000, peers' up probability=0.3, and query model: *Zipf.* y-axis means the satisfied QoA in average. For (a) and (b) x-axis means the number of query counter, while for (c) and (d) it means simulation time slot.

original content. As Figure 5 shows, the *Proportional* scheme offers higher satisfied QoA than the *Uniform* scheme for the *Zipf*-like access query model.

Table 4: Simulation parameters and their value ranges for the simulation runs with the dynamic placement.

Parameters	Values
peer up probability	0.0 - 0.9
content popularity	.01 - .99
number of peers	100, 1000
number of origin contents	1000
number of query events	1000
number of simulation time slots	100
range of demand availability values	.50 - 0.99
range of supply data availability	.50 - 0.99
query distribution	Uniform, Zipf

Table 4: Simulation parameters and their value ranges for the simulation runs with the dynamic placement.

Parameters	Values
proactive placement heuristics	Random, UP, HA, HA+UP
on-demand placement heuristics	Local-LRU, UP, HA, HA+UP
test graphs	G1(100,300), G2(1K,5K)

4.4.2 Effects of Placement Schemes on Satisfied QoA

In the second 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 6 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 6(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.

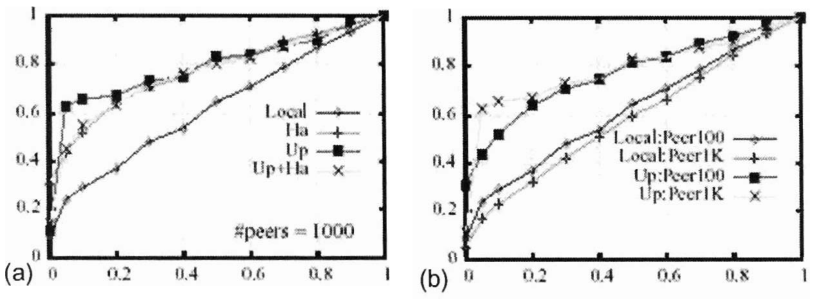

Figure 6: Effect of placement strategies on satisfied QoA where proactive placement: *Random* and peers' up probability=0.3. (a) average satisfied QoA from all four heuristics used. #peers=1000, (b) a comparison of the average satisfied QoA between *Local-LRU* and *UP* heuristic with different graph sizes. The number of peers of Peer100 and Peer1K is100 and 1000, respectively. X-axis means replication ratio, 0-100%, while y-axis means the satsfied QoA

4.4.3 Satisfied QoA versus Hit Probability

Maximizing hit probability is one frequently used goal for content replication [13]. In Figure 7 we show a comparison between the two replication 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 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 at 0.3 and we used *Random* and *UP* placement schemes for proactive and on-demand phase respectively. As Figure 7 shows, satisfying required QoA incurs higher cost, i.e. more number of replicas than just maximizing hit probability. For example, when the replica rate is 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.

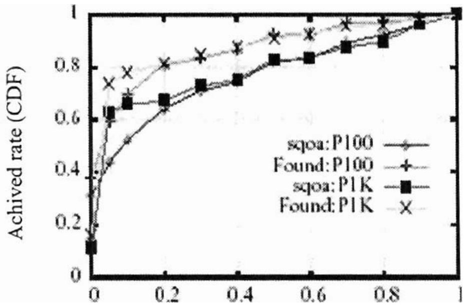

Figure 7: Comparison of replication cost for different replication goals: satisfying QoA vs. maximizing hit probability. P100 and P1K mean 100 and 1000 nodes, respectively. X-axis means replication ratio.

4.5 Discussion

4.5.1 Summary of Simulation Results

The following observations could be identified from our experiment results:

- The location of replicas is a relevant factor for the availability QoS. Even though the QoA improvement could be achieved by increasing replica numbers, replicas' placement and their dependability affected the QoA more significantly.
- Using a heuristic method is more efficient than the exact method, at least in terms of the runtime complexity, to find a good placement for large graphs. But, the replica degree of their placement results are in most cases higher than those of exact methods. Furthermore, the heuristics give no guarantee for availability QoA.
- In opposite to the heuristic method, the exact method can exactly find the optimum and give QoA guarantee with its placement results, although the runtime complexity is very high: to find the optimum, we need to call the state enumeration method $2^{|V|}$ times, i.e., for all the possible replica solution sets. The algorithm complexity is then $O(2^{|V|} \cdot 2^{|V|+|E|})$ to find the optimum with all possible instances.
- Satisfying availability QoS requires more replicas than only improving performance, e.g. increasing hit rate.

4.5.2 Algorithm Improvement: Admission Controlled Placement for Guaranteeing QoA

We investigate for new placement algorithms which reduce the exponential runtime complexity while guaranteeing QoA. One of possible solution algorithms is using the admission control technique [2] where the placement is controlled based on available resources of the system nodes and links: each system node (peer) checks its current (node and intrinsics) availability and either accepts or rejects the new request based on the check result. For this purpose, we take additional constraints on resource capacities, e.g. storage space, load and access bandwidth capacity. We further assume that there may be at least one replica or its original data available at any access time. In the new placement algorithm which we call *admission-controlled placement*, the replica placement can be performed in two (or more) phases: *Base placement* and *Optimization*. In the base placement phase, we find the highest available path for each of demanding nodes and test whether the path gives a QoA guarantee, i.e. whether the supplied QoA achieved by the selected path is greater than the demanding QoA for the node. If the test fails, we select a node along the path, which is closest to the destination node (i.e. service supplying node) and has enough resource to be allocated and fulfils the QoA requirement. The node is then added to the replica node set. After the first placement phase, this replica node set will then give a QoA guarantee for all demanding nodes.

The resulting replica set R guarantees QoA for all demanding nodes. However, R is neither the optimum nor has been optimized. Thus, the second phase of our admission-controlled placement is mainly to optimize the placement. We try to reduce |R| determined in the first phase, while keeping the QoA guarantee. One simple approach is to delete the replicas with lower supply QoA values. For example, each replica node of R is taken (or hidden). We then check for the demanding nodes which are assigned to the hidden node whether their demand QoA values can be fulfilled by supply QoA of all other replica nodes in R. If yes, the hidden node can be deleted. This test is repeated for all the replica nodes of R. We call this phase as *Optimization* phase.

The optimized placement R2, which is determined in the second phase of the admission-controlled algorithm, offers also QoA guarantee for all demanding nodes. However, it may still not be the optimum. Therefore, one may adopt additional optimization techniques such as '*Move and Update*' [21]. We used the admission-controlled algorithm for both static and dynamic replication modes. Currently, we are collecting simulation results of the algorithm. [19] shows a pseudo-code the base placement module of the admission-controlled algorithm.

5 Related Work

The key ideas on which our work on QoA concept in this paper bases are (i) an availability-centric view on QoS and (ii) satisfying different levels of QoA values required by individual users. Since the common goals associated with replica placement problems in existing studies are reducing clients' download time and alleviating server load, the main feature of the problem solving approaches for this problem category is that they usually addressed the cost and resource minimization issues, but not the question how to guarantee the required availability.

Kangasharju et al [14] studied the problem of optimally replicating objects in content distribution network (CDN) servers. As with other studies [15-17], the goal of their work is to minimize object lookup time/cost, i.e., minimize average number of nodes visited to find the requested object. Furthermore, they assumed that all of the objects are always available in their origin server, regardless of the replica placement.

In [18] Kangasharju et al. also studied the problem of optimally replicating objects in P2P communities. The goal of their work is to replicate content in order to maximize

References

[1] Zheng Wang. *Internet QoS: Architectures and Mechanisms for Quality of Service.* Lucent Technologies, 2001.

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

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

[4] HP Forum. *Providing Open Architecture High Availability Solutions*, Revision 1.0, Feb. 2001. document available at <http://www.mcg.mot.com/us/products/solutions/ha_solutions.pdf>

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

[6] 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.

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

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

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

[10] G. On, J. Schmitt and R. Steinmetz. "On Availability QoS for Replicated Multimedia Service and Content." in *LNCS 2515 (IDMS-PROMS'02)*, pp. 313-326, Portugal, Nov. 2002.

[11] Christos H. Papadimitrio and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, ISBN 0-13-152462-3, 1982.

[12] C. Lucet and J.-F. Manouvrier. "Exact Methods to compute Network Reliability". In *Proc. of 1st International Conf. on Mathematical Methods in Reliability*, Bucharest, Roumanie, Sep. 1997.

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

[14] J. Kangasharju, J. Roberts and K.W. Ross. "Object Replication Strategies in Content Distribution Networks", *Computer Communications,* Vol.25 (4), pp. 376-383, March 2002.

[15] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt and L. Zhang. "On the Placement of Internet Instrumentation", In *Proc. of IEEE INFOCOM'00*, Mar. 2000.

[16] 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.

[17] P. Krishnan, D. Raz and Y. Shavitt. "The Cache Location Problem", In *IEEE/ACM Transactions on Networking, 8(5)*, pp. 568-582, Oct. 2000.

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

[19] G. On, J. Schmitt and R. Steinmetz. "Admission Controlled Replica Placement algorithms." Technical Report KOM-TR-2003-7, April. 2003.

[20] 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.

[21] N. Mladenovic, M. Labbe and P. Hansen. "Solving the p-Center Problem with Tabu Search and Variable Neighbourhood Search" July 2000. paper available at <http://www.crt.umontreal.ca/>

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 5, maximizing hit probability does not satisfy the required QoA and, furthermore the two different goals lead to different results.

Lv et al. [6] and Cohen and Shenker [7] 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 [20] 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 address an availability guarantee (guaranteedQoA = 1 at least), whereas finding the optimum in an exact way is one of the focus of this paper.

6 Conclusion

We took an availability-centric view on QoS and focused on the issues of providing models and mechanisms to satisfy availability requirement for widely distributed systems such as P2P systems. We developed a concept called *quality of availability (QoA)* in which the availability is treated as a new controllerable QoS parameter. Based on the QoA concept, we modelled widely distributed systems as a stochastic graph where all nodes and edges are parameterized with known availability and failure probabilities.

We tackled specifically the replica placement problem in which we specified different placement problems with different QoA metrics such as *satisfiedQoA* and *guaranteedQoA*. Our goal was choosing the number and location of replicas to satisfy the availability QoS requirement for all individual peers, while taking intermittent connectivity of service systems explicitly into account.

From simulation studies, we have learned that

- heuristics cannot give any guarantee on their achieved availability QoS, even when hey achieve reasonably high availability QoS,
- the location of replica is a more relevant factor than its number for satisfying the required QoA,
- in opposite to the heuristic method, the exact state enumeration algorithm guarantees the availability QoS with its placement results, although the algorithm has an exponential runtime complexity, and
- satisfying availability QoS requires more replicas than for only increasing the performance.

Our proposed QoA concept 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 dynamic replica placement for availability QoS guarantees. Furthermore, for a practical use of our proposed model, one 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.

[redacted]

Kevin Jeffay
Ion Stoica
Klaus Wehrle (Eds.)

[redacted]

LNCS 2707

Quality of Service – IWQoS 2003

11th International Workshop
Berkeley, CA, USA, June 2003
Proceedings

Springer