On Availability QoS for Replicated Multimedia Service and Content

Giwon On, Jens Schmitt, Ralf Steinmetz
Multimedia Communications (KOM), Darmstadt University of Technology, Germany
Tel.: +49-6151-166150, Fax: +49-6151-166152
{Giwon.On, Jens.Schmitt, Ralf.Steinmetz}@kom.tu-darmstadt.de

Abstract 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. In this paper, we take an availability-centric view on QoS and focus on the issues of providing availability guarantees for distributed and replicated multimedia services and contents. For this purpose, we develop a concept called *quality of availability* (QoA) in which the availability is treated as a new controllable QoS parameter. We especially tackle the replica placement (RP) problem and study the effects of number and location of replicas on the achieved availability. We decompose the RP problem into three sub-problems: (1) finding a "good" placement for a fixed number of replicas, (2) checking the reached QoA with a selected replica placement, and (3) determining the number and location of replicas for satisfying a requiredQoA. For each RP sub-problem, we review existing related work for algorithms ranging from heuristic to exact methods as well as devise new algorithms where existing work is lacking and evaluate their achieved QoA. Based on a simulation study, we find that (1) the location of replica is a more relevant factor than their number for satisfying the QoA requirements by different users, and (2) the heuristic methods, in general, cannot give any guarantee for their achieved QoA, even though they are very efficient for large size graphs. Our proposed QoA concept and model can be used as a base mechanism for further study on the availability and reliability QoS for dynamically changing network service environments.

Keywords: multimedia service and content replication, availability, QoS, quality of availability (QoA), replica placement problem, stochastic graph

1 Introduction

1.1 Satisfying Availability - the Key for Successful Services in Internet

The rapid popularization of Internet-based distributed services and applications have inspired the research and development of technologies for providing resource assurance and service differentiation which have motivated the development of the concept, architecture and mechanisms of *quality of service* (QoS). The main focus of QoS is improving the service quality by addressing the issues of resource management, service differentiation and performance optimization [1,2].

Even though there are many significant research results, technology advances and solutions in QoS since the last 20 years, their application to commercial products or systems was not so successful, in comparison to their attention in the research arena. One critical reason probably is that, as H. Schulzrinne pointed out in [3] and an interview statistic mentioned in [4], the main research focus for QoS was to control transmission characteristics like bandwidth, delay, and loss. This is because applications on the Internet typically assumed in need for QoS support, such as video conference, video-on-demand, tele-teaching/learning, 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, the availability 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.

1.2 Differentiation of Service Classes and Availability Requirements

While most research efforts in high availability and fault-tolerant systems areas have their focus on achieving the so-called 'five nines' (99.999%) availability [5], there is a demand for service differentiation from service consumers and providers due to costs and competitive nature of the marketplace, which derives for the mechanisms that support different levels of services and their availability. In fact, the need for service

differentiation can be observed from different services, such as video-on-demand (VoD) or news-on-demand over the Internet, as wells as different users' requirements which depend on the service type they demand, on the service time when they access, on the peripherials they have, and on the service price they pay.

From the service system provider's point of view, on the other side, not all system components need to have the same redundancy level, i.e., availability level to offer. The availability level required for individual system components depends on the fact of how much they should be reliable and are critical for offering the service. For example, in developing replication mechanisms for increasing availability of services and their data in a distributed multimedia system *medianode* [6], we analysed the characteristics of multimedia contents and their meta-data and could identify that not all service operations and not all data access functions require the same availability level of the 'five nines'.

1.3 The Main Focus and Approach

The work in this paper is strongly motivated by the two aspects mentioned above. So, the main focus is building a model and mechanisms to study the problem of how to satisfy and guarantee different availability requirements for distributed and replicated multimedia services in a wide-area internetwork like the Internet, 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 (RP) problem and study the effects of number and location of replicas on the reached availability QoS. For this purpose, we develop a concept called *quality of availability* (QoA) in which the availability is treated as a new controllable QoA parameter. Based on the QoA concept, we model the distributed multimedia system as a stochastic graph where all nodes and edges elements are parameterized, statistically independently of each another, with known availability and failure probabilities. An availability requirement value is additionally assigned to each node so that the target replica placement problem is to find replica set with which the availability requirements for all nodes are satisfied. We decompose the RP problem in three questions: (1) finding a "good" placement for fixed number of replicas, (2) checking the reached QoA with a selected replica placement, and (3) determining the number and location of replicas for satisfying a required QoA with absolute guarantees. Thus, the main focus of the paper is not developing an additional, new algorithms for the RP problem, but instead specifying the availability QoS concept and model. For each RP question, we review some solution algorithms from heuristic and exact methods and evaluate their achieved QoA. Based on a simulation study, we find that (1) the location of replica is a more relevant factor than its number for satisfying the required availability QoS, and (2) the heuristic methods do not give any guarantee for their achieved QoA, even though they are very efficient for large size graphs. Note that 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 problem.

1.4 Outline

The rest of this paper is organized as follows. In Section 2, we develop the concept of quality of availability and describe the QoA metrics to be used in this work. Section 3 presents the replica placement problem and details the algorithms that we reviewed and modified for our problem. In Section 4, we present our implementation methods including the experimental simulation environment and in Section 5 we evaluate the results. Section 6 discusses related work. Finally, Section 7 concludes the paper.

2 The Concept of Quality of Availability (QoA) and its Model

2.1 Basic Idea

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 amount in which the required service functions as expected and its 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 in-

terested 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 and presents our methodology to study the QoA.

2.2 The QoA Metrics and Parameters

A service is said to be *available* when it functions as expected and its data is reachable. Commonly we distinguish between two levels of available services: *basically available* (BA) and *highly available* (HA). At the BA level, a service delivers correct functionality as long as no faults occur, but it neither offers any redundancy for its system components and data, nor fault detection and recovery mechanisms, while a HA service, in addition to the BA level's feature, provides a certain level of redundancy and eventually the mechanisms for fault-tolerance support [5,11].

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 [11]. 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 like the cases in network QoS [2,7]:

- *deterministic* - a service (or its data item) is reachable all the time with an availability guarantee of 99.99 percent. This means for a service that the time duration where the service is unreachable should be absolutely 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.
- *time average* - the available service time (or data access time) is more than 8000 hours over a year (8760 hours).

Actually, the exact form of QoA parameter 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(v): $QoA_s(v)$ = satisfied availability(v)/required availability(v), $\forall v \in V_R$ *with* V_R *(V without R)*
- minQoA: $QoA_{min} = \min \{ QoA_s(v) : \forall v \in V_R \}$
- avgQoA: $QoA_{avg} = 1/n(\sum QoA_s(v))$ with $\forall v \in V_R$ and $n = (|V| - |R|)$
- guaranteedQoA: $QoA_g = |V_s|/|V|$, V_s *= set of nodes with* $QoA_s(v) \geq 1$

Beyond the QoA parameters mentioned above, we can also take into account typical QoS parameters like delay, jitter and data loss. Especially, taking the delay as QoS parameter into account makes the QoA model more realistic, because service consumers usually expect to receive a service within an acceptable response time, when they require the service, and when it is assumed that the service is available. One other QoA parameter can be the consistency parameter which can be modeled either as staleness or the total time duration of data being inconsistent.

2.3 Replication as Concept for Increasing Availability

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. 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 depend only on the user's access patterns, but also on the available, continuously changing, but limited system and network resources and on the costs which service (or content) providers need to pay for satisfying a required availability level.

3 The Replica Placement Problem

3.1 Notations and Problem Formulation

Distributed, networked service systems that consist of storage/server 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 between 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 $R \subset V$. Concerning the RP problem, the following sections introduce three sub-problems that are explored in this work. For all three sub-problems, we model the RP problem as a *static, stochastic and unconstrained* graph. Table2 summarizes the notation used for formulating and specifying the RP problem throughout this work.

Notation	Description
$G(V, E)$	*a graph with V, the set of nodes, and* $E \subset V \times V$ *, the set of links*
D	*set of service demanding nodes and a subset of nodes of G,* $D \subset V$
S	*set of service supply nodes and a subset of nodes of G,* $S \subset V$
R	*set of replica nodes, a subset of nodes of G,* $R \subset S \wedge R \subset V$
$\|V\|, \|D\|, \|S\|, \|R\|$	*cardinality of the node sets V, D, S, R*
$T(R)$	*topological placement of R, i.e., the location of R's elements*
p_e	*availability probability of the element e (* $e \in V$ *or* $e \in E$ *) for service supply*
q_e	*failure probability of the element e (* $e \in V$ *or* $e \in E$*) for service supply,* q_e *=1-* p_e
pd_e	*required service availability of service demanding node e (* $e \in V$ *)*
x_e	*a boolean variable for the state of the component e (* $e \in V$ *): 1 if e functions else 0*
$a(G)$	*availability of G*
g_i	*state of G*
$G(g_i)$	*partial graph of G associated with* g_i
$A_e(R)$	*achieved QoA for a service demanding node e of V with R: 1 if satisfied else 0*
$\overline{A}(R)$	*achieved QoA for all service demanding nodes of G with R: 1 if satisfied else 0*

Table 1: Notation.

3.2 Finding a "Good" Placement for a Fixed Number of Replicas

As input, a stochastic graph *G (V, E) is* given. As the second parameter, a positive integer number *k* may also be given. 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|, T(R), a)* is satisfied for given availability requirement for service demanding nodes. How well the optimization condition is satisfied depends on the size of |R| and the topological placement T(*R)*. Because the main goal associated with placing replicas on given networks in our work is satisfying availability QoS which can be required in different levels, we take the availability and failure parameters as our key optimization condition, i.e., *O(|R|, T(R), ReachedAvailability).* 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|, T(R), 1.0), O(|R|, T(R), .90), O(|R|, T(R), Avg),* respectively. For these conditions, the replica set *R* must be chosen such that the maximum, average or any given failure bound for service and its data access meets the QoA requests for any demanding node (client node) of *V*.

The RP problem can be classified as NP-hard discrete location problem. 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*, *Tabu Search*, etc. are applied to many location problems and have shown their efficiency [15,16,17,18,19,21,22]. In this work, we take some basic heuristic algorithms as follows. But, different variants of these heuristics and any such improvement can be used with light modifications to enhance the efficiency and performance of our basic heuristics:

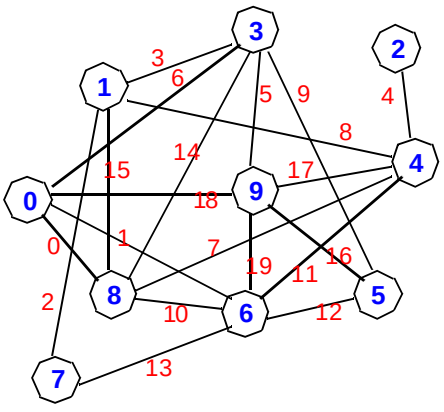
- *Random (RA)*. By using a random generator, we pick a node *v* with uniform probability, but without considering the node's supplying availability value, and put it into the replica set. If the node already exists in the replica set, we pick a new node, until the given number reaches *k*.
- *HighestFirst (HF)*. For each node *v*, we calculate v's actual supplying availability value (in math. form here) by taking the availability values of all adjacent edges of the node into account. The nodes are then sorted in decreasing order of the actual availability values, and we finally put the best *k* nodes into the replica set.

```
Algorithm HF+TR
Input:              a parameterized graph G (V,E) and a number k;
Output:             a replica node set R
Initialization:     replica node set R = candidate node set candSet = {}

1. forall nodes v of V
2.  forall adjacent edges of v,
3.  calculate max/min/mean values for demanding and supply availability values
4. forall nodes v of V
5.  build a candSet with nodes of which their availability value and degree are
          greater than the average values, respectively.
6.  sort the nodes in candSet in decreasing order of the supply availability value
7.  for nodes which have the same availability values,
8.         sort these nodes in decreasing order of the (in/out) degree
8. while (|R| <= k)
9.  put the best k nodes from candSet into R
10.return R
```

Figure 1: Pseudo-code of the *HF+TR* algorithm for finding a placement

- *TransitNode (TR)*. The basic principle of the *TransitNode* heuristic 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 decending order of (in/out) degree. This is



NodeId	0	1	2	3	4	5	6	7	8	9
Req.QoA	94	92	91	92	93	94	93	92	95	95
Sup.QoA	96	95	91	96	98	94	95	92	96	97

EdgeId	0	1	2	3	4	5	6	7	8	9
Fail-Prob	3	9	1	6	3	1	2	4	4	2
EdgeId	10	11	12	13	14	15	16	17	18	19
Fail-Prob	6	3	8	1	3	7	5	2	4	8

Figure 2: An example stochastic graph: the demanding and supplying availability values for all nodes are independent and decoupled. To each edge a failure probability value is assigned. The values are in percentage.

due to the observation that nodes in the core of the Internet that act as transit points will have the highest (in/out) degrees [15,16].

- *HighestFirst+TransitNode (HF+TR)*. This method is a combination of the *HF* and *TR* algorithms. Figure 1 detail the *HF+TR* algorithm.

We applied the *HR+TR* algorithm to an example stochastic graph which represents a model of a virtual network topology (Figure 2). In this example graph G(V,E) with |V|=10 and |E|=20, nodes and links are parameterized with randomly generated probability values, e.g., the demanding and supply availability values of nodes are between 90% and 99%, and the failure probability values of links are between 1% and 10%. For a given number $k = 1$, the candidate replica nodes are {4}, {9}, {8}, {3} and {0}. By considering only the availability value (*Sup.QoA*), the algorithm choices {4} as the replica set.

3.3 Calculating the Reached QoA

Given a stochastic graph *G(V,E)* and a replica set *R* with its topological placement T(R). The objective of this problem is to check for all demanding nodes whether the reached availability satisfies the required QoA for them, i.e., whether $\bar{A}(R)$ is 1 or 0. In comparison to the problem of finding a good placement decribed in Section 3.2, this problem requires a solution which exactly tests whether the result is 1 or 0. Some similar works are introduced in the literature, which are devoted to the problem of *network reliability* [13]. The methods that provide an exact reliability are called exact methods, in contrast to the heuristic methods which provide an approximate result.

Enumerating all possibilities without skipping any solution case requires to take exact methods for solving this problem. From some exact methods which are proposed in the literature, we adopted the state enumeration method[13] 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 the replica set given as input. Figure 3 details the *StateEnumeration* algorithm.

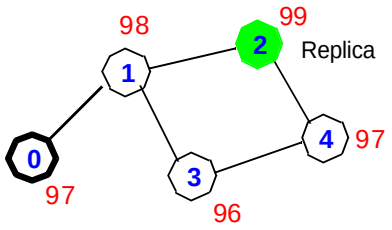
```
Algorithm StateEnumeration
Input:    a parameterized graph G (V,E) and a placement (replica set R);
Output:   the QoA: either 1 (satisfied) or 0 (not satisfied)
Initialize variables:state matrix

1. for all nodes v of V/R
2. build a state martix for all partial graphs of G
3. for all states
4.  check the availability of the state, i.e., exist at least a path?
5. for the states for which the checked availability value is 1,
6.  compute availability probability for each state
7. sum all the availability probability values of satisfied states
8. if p_a(v) <= A_v(G) then A_v(R) =1
9. for all nodes v,
10. if A_v(R) =1
11. then  Ā(R) = 1, i.e., the QoA of G with R is satisfied
12. else  Ā(R) = 0
13.return QoA
```

Figure 3: The *StateEnumeration* algorithm for calculating the reached availability, QoA

We applied the *StateEnumeration* algorithm to an example stochastic graph G(V,E) with |V|=5 and |E|=5, placement R={2}, |R|=1, as shown in Figure 4. As test node, we take the node 0 which has the availability requirement value 97%. Figure 4 (right) presents the state matrix, the availability and the sum of availability probability value for each partial graph. In this example, we only encounter the states of nodes to reduce the time complexity, i.e., from $2^{|V|+|E|}$ to $2^{|V|}$. The first column means number of the partial graphs to be tested. Instead of considering all partial graph cases of $2^{|V|+|E|}$, we only have the half size of them by skipping the cases in which the node 1 is 'not available', i.e., (the state value of the node 1 is 0), because the node state 'zero' of the node 1 causes no further possible connection for the test node 0 to build any path to the replica node 2. After building a state matrix for the nodes 1,2,3 and 4 at the second column, we check whether there is any path between the node 0 (test node) and the node 2 (replica node) at the third column. According to the result of this check, we calculate the availability probability values for each partial graph of which the

availability check value is 1: as shown in the fourth column, we calculated the *Pr(G_m)* just for the first 4 partial graphs (m = 1,2,3 and 4). The summation of the availability probability values of the four satisfied states is: *A_node0(G) = Pr(G_1) + Pr(G_2) + Pr(G_3) + Pr(G_4) = 0,97135624*, and this availability value is greater than the availability value required by the node 0. Indeed, the QoA for the node 0 is: *A_node0({2}) = 1.*



m	StateMatrix	a(G_m)	Pr(G_m)
1	*1 1 1 1*	*1*	*0,98 * 0,99 * 0,96 * 0,97*
2	*1 1 1 0*	*1*	*0,98 * 0,99 * 0,96 * 0,03*
3	*1 1 0 1*	*1*	*0,98 * 0,99 * 0,04 * 0,97*
4	*1 1 0 0*	*1*	*0,98 * 0,99 * 0,04 * 0,03*
5	*1 0 1 1*	*0*	*/*
6	*1 0 1 0*	*0*	*/*
7	*1 0 0 1*	*0*	*/*
8	*1 0 0 0*	*0*	*/*

Figure 4: An example graph G=(V,E), |V|=5, |E|=5 (left). Only the node states are enumerated and the cases where the node 1's state is 0 are skiped (right).

3.4 Determining the Number and Location of Replicas for Satisfying Required QoA

As input, only a stochastic graph *G(V,E)* is given. It has to be determined (a) how many replicas must be deployed and (b) where these replicas should be placed to guarantee the required QoA.

Satisfying a certain, required QoA value with a guarantee means that we have to offer always a replica set which fulfils the given QoA requirements in any case. Heuristics are not proper approaches for solving this problem, because they give not always a solution with QoA guarantee. To solve this problem, we generally can use the exact methods, one possible case may be enumeration method which is described in the subsection 3.3. For solving this third problem, 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 $O(2^{|V|} \cdot 2^{|V|+|E|})$.

4 Simulation Environment

We built an experimental environment to perform a simulation study for the three RP sub-problems addressed in Section 3. Our goal in conducting an availability evaluation is to determine exactly the replica set *R* for given QoA requirements, and to study the effect of changing |*R*| and *T(R)* on the required and reached QoA which is given as the optimization condition *O(|R|, T(R), a)*, on the other side. For our availability evaluation, we conducted simulations on random network topologies.

By using Leda graphic library [24] several random topologies in different sizes can be generated at run time. We also used graph files which are generated by the topology generator Tiers[25]. To reflect the actual Internet topology, we used several different network trace data from NLANR[26], which describe the Autonomous System(AS) topology on a different day, and associated in/out-degree of the AS with each node of the randomly generated graphs. But, because we had no access to any real data concerning the availability or the failure parameters, we simply assign the values from a certain range to each node and each edge.

The availability and failure probability parameters for nodes and edges of the graphs are single values: for example, 50, 80, 90 or 99% as availability values and 10, 5, or 1% 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 (client node), then its demanding availability value is used, while for supplying node the actual supplying availability value is, for example, calculated by multiplicating the availability values of its supplying 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 *HA+TR* algorithm, a node should have a high supplying availability value.

As replication model, we assume the *full replication* in which the whole data items of an origin server system are replicated to other nodes located within the same network. *Mirroring* is a typical case of the full replication model. The simulation program is written in C/C++ and runs on Linux and Sun Solaris 2.6 machines.

5 Evaluation of Reached QoA

In this section, we present our experiment results. We evaluated the reached QoA of our heuristics and the exact enumeration method using topologies of different sizes. We ran each basic heuristic and the exact state enumeration method on each topology using different value ranges for the availability and failure probability parameters of nodes and edges. The demanding and initial supplying availability values of the nodes and the failure probability values of the edges are assigned randomly, from a uniform distribution where we varied the parameter values as Table 3 shows. To evaluate the QoA offered by our heuristics and StateEnumeration, we used the QoA metrics defined in Section 2.2.

Type	Parameter	Notation	Value
Graph	node and edge size	$\|V\|:\|E\|$	*G1(20:30), G2(100:300)*
Edge	edges' failure probability	q_l	*1 ~ 10 %, 0%*
Node	nodes' required availability	p_{nReq}	*90~99%, 50~99%, 50-90-99%*
	nodes' supply availability	p_{nSup}	$p_{nSup} \geq p_{nReq}$

Table 2: QoA parameters and their values for our simulaiton

5.1 Relative Comparison of Reached QoA by Heuristics

We evaluate at first the reached QoA by our simple heuristics. 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 reached QoA of each heuristic to this baseline and present the relative QoA improvement obtained with each heuristic.

5.1.1 Effects of |R| and T(R) on Reached QoA

The first experiment was 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) $QoA_{min} > 0.9, 0.95,$ and 0.99, respectively, and (2) $QoA_{avg} > 1.0$. In this case, there was no constraints on the topological location of the replicas, i.e., $S = V$ and replicas may be placed at any node v in G.

Figure 5 (*left*) 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 5, we can see that our heuristics *HA* and *HA+TR*, although they are very simple, reach significantly higher QoA in comparison to the baseline placement. 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.

5.1.2 Effects of Varying Availability Requirement Value Ranges on QoA

In the second experiment, to study the effects of difference ranges of the required availability values on the reached QoA, we varied the ranges of required availability values (p_{nReq}) from 50~99% to 80~99% and 90-99% for the same graph G2. We took also p_{nReq} as different single values like 50-80-99%. As Figure 5 (*right*) shows, the improvement rate of the reached QoA is better when the p_{nReq} is distributed in wider ranges.

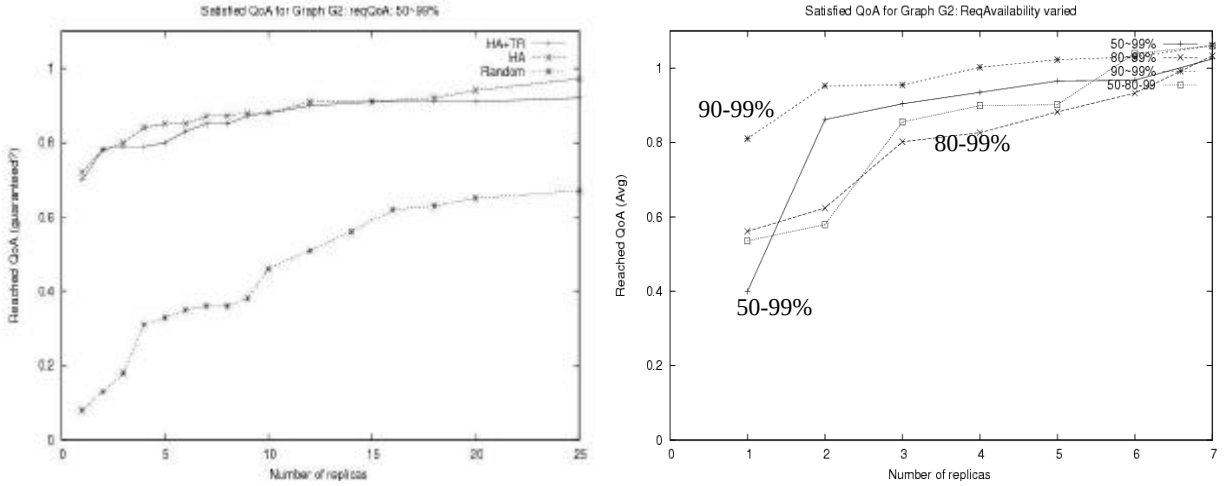


Figure 5: Reached QoA values by our hueristics: (*left*) different heuristics, (*right*) different ranges for required availability values

5.2 Exact Evaluation of Reached QoA by Heuristics

We now evaluate the QoA reached by our heuristics in an exact form and check whether the reached QoA can really satisfy the required QoA for all demanding nodes (*QoA guarantee test*). We test also how many replica nodes do the heuristics need to give a QoA guarantee. For this purpose we ran our *StateEnumeration* routine described in Figure 3 with replica sets produced by our heuristics *HA* and *HA+TR* as input. Due to the exponentially growing runtime complexity and the memory requirements with growing graph sizes, we limited our experiments for the *StateEnumeration* to a small graph, the test graph *G1* with |V| = 20 and |E| = 30. Table 3 shows the detailed test result from *HA+TR*. For the calculation of the average (*AvgQoA*) and minimal reached QoA (*MinQoA*), we excluded the QoA values for replica nodes.

No. of replicas, *R*	Replica locations, *T(R)*	Avg QoA	Min QoA	% QoA Guarantee	% QoA by exact test
1	8	**1.0118**	0.9100	0.75	**0.80**
2	8,10	1.0194	0.9100	0.75	not checked
3	8,10,12	1.0226	0.9193	0.75	not checked
4	8,10,12,11	1.0355	0.9496	0.85	not checked
5	8,10,12,11,13	1.0399	0.9496	0.85	not checked
6	8,10,12,11,13,0	1.0487	**0.9591**	0.90	**0.95**
7	8,10,12,11,13,0,16	1.0556	**0.9900**	0.95	**1.00**
8	8,10,12,11,13,0,16,1	1.0577	0.9900	0.95	not checked
9	8,10,12,11,13,0,16,1,2	1.0610	0.9900	0.95	not checked
10	8,10,12,11,13,0,16,1,2,5	1.0711	1.0000	**1.00**	**1.00**

Table 3: A detailed result for *HA+TR* with *G1*, failure probability: *0%*, and req. availability range: *90-99%*

Even though *HA+TR* could reach the average QoA (1.0118) greater than 1 with one replica node, it could not offer the QoA guarantee: 10 replicas were needed to satisfy the QoA guarantee for the small graph.

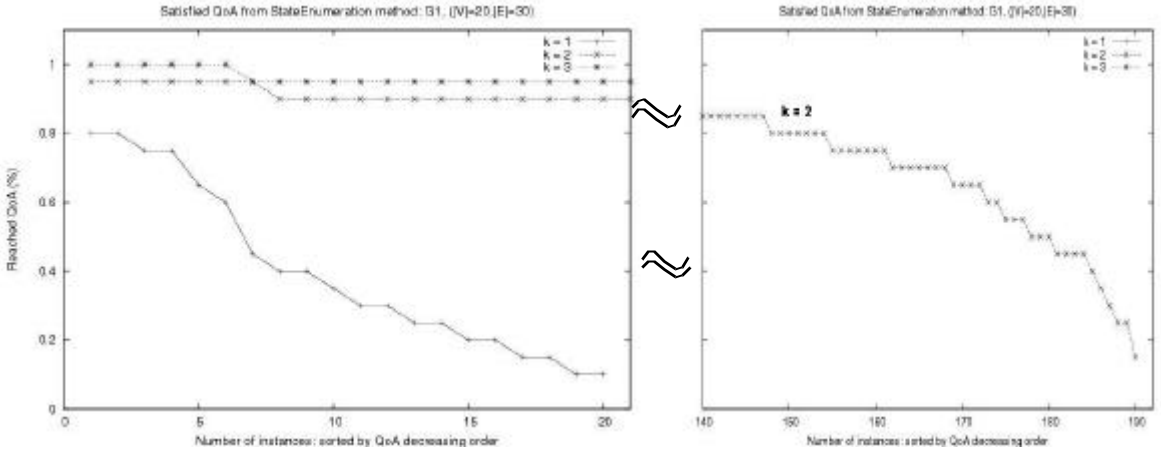


Figure 6: Reached QoA that was checked exactly by *StateEnumeration*

5.3 Finding the Optimum - |*R*| and *T(R)*

In the last experiment, we considered the case of finding the optimum, i.e., the minimal number of replicas and their geographical placement which satisfies the availability QoS with guarantee. We re-used *StateEnumeration* and the test graph *G1* with the same values for availability and failure probability parameters. 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 $gQoA = 1.0$ (a QoA with guarantee). Table 5 shows the reached QoA values at each *k (k=1,2,3).* Figure 6 plots the reached QoA that *StateEnumeration* algorithm calculated exactly with each instance for the given *k.* The wider spectrum of the left part is for highlighting the reached QoA from all of the instances for k=1 and the choosen instances for k=3. The right part of Figure 6 shows how the reached QoA varies in the case of k=2, and how big is the gap between good and bad QoA rates reached by the instances.

No. of replicas	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	not checked	not checked	{0,11,16},{0,16,18},{8,11,16},{8,16,18},{11,12,16} ,{11,13,16}

Table 4: A test result from *StateEnumeration* with *G1*, failure probability: *0%*, and req. availability range: *90-99%*

5.4 Discussion

The following observations could be identified from our experiment results: (1) 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; (2) 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.; (3) in opposite to the heuristic method, the exact method guarantees the availability QoS with its placement results, although the runtime complexity is very high: $O(|V_R| \cdot 2^{|V|+|E|})$ and $O(2^{|V|} \cdot |V_R| \cdot 2^{|V|+|E|})$ for the availability checking and the guaranteed QoA problems, respectively.

6 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 placement problems in exsisting 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. Furthermore, we can find an "good" upper bound, if the selected placement meets the required availability QoS, but it is not guaranteed that the selected placement always meets the availability requirement.

7 Conclusion

In this paper we took an availability-centric view on QoS and focused on the issues of providing models and mechanisms to satisfy availability requirement for replicated multimedia services and contents. 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 a distrbuted multimedia system as a stochastic graph where all nodes and edges are parameterized with known availability and failure probabilities. We especially tackled the replica placement problem in which we specified different placement problems with different QoA metrics such as *MinQoA*, *AvgQoA*, and *GuaranteedQoA*. The primary result shows already that (1) the location of replicas is a more relevant factor than their number for satisfying the availability QoS for different users, and (2) the heuristic methods could not give any guarantee for their achieved QoA, even though they are very efficient for large size graphs. Our proposed QoA concept and model can be used as a base mechanism for further study on the availability and reliability QoS with dynamic replication problems and mobile storage planning problems. We are currently extending the simulation model to build a more realistic QoA model by taking QoS and consistency parameters into account.

References

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

[2] J.Schmitt. *Heterogeneous Network QoS Systems*. PhD thesis, Darmstadt University of Technology, December 2000.

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

[4] SEQUIN project. "Quality of Service Definition" *SEQUIN Deliverable D2.1,* April 2001. available from <http://www.dante.net/sequin/QoS-def-Apr01.pdf>

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

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

[7] V. Firoiu, J.-Y. Le Boudec, D. Towsley, and Z-L Zhang. "Advances in Internet Quality of Service", available from <http://dscwww.epfl.ch/en/publications/documents/tr00_049.pdf>

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

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

[10] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, ISBN 0-201-05594-5, 1984.

[11] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems*. 3rd Ed., Chapter 1, 8, 14 and 15, Addison-Wesley, 2001.

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

[13] 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.
[14] Pierre, I. Kuz, M. van Steen and A.S. Tanenbaum. “Differentiated Strategies for Replicating Web documents”, In *Proc. of 5th International Workshop on Web Caching and Content Delivery*, Lisbon, May 2000.
[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 *IEEEEE/ACM Transactions on Networking, 8(5)*, pp. 568-582, Oct. 2000.
[18] J. Kangasharju, J. Roberts and K.W. Ross. “Object Replication Strategies in Content Distribution Networks”, In *Proc. of 6th Web Caching and Content Distribution Workshop*, Boston, MA, USA, June 2001.
[19] J. Chuang. “Resource Allocation for stor-serv: Network Storage Services with QoS Guarantees”, In *NetStore’99, Seattle WA*, October 1999.
[20] Haifeng Yu and Amin Vahdat, "The Costs and Limits of Availability for Replicated Services." In *Proc. of the 18th Symposium on Operating Systems Principles (SOSP)*, October 2001.
[21] I. Cidon, S. Kutten, and R. Soffer. “Optimal allocation of electronic content”, In *Proc. of IEEE INFOCOM’01*, Anchroage, AK, April 2001.
[22] L. Qui, V.N. Padmanabhan, and G.M. Voelker. “On the placement of web server replicas”, In *Proc. of IEEE INFOCOM’01*, Anchroage, AK, April 2001.
[23] M. Nicola and M. Jarke. “Performance Modeling of Distributed and Replicated Databases”, *IEEE Trans. on Knowledge and Data Engineering*, Vol.12(4), pp.645-672, July/Aug. 2000.
[24] Leda - the library of efficient data types and algorithms. Algorithmic Solutions Software GmbH. available from <http://www.algorithmic-solutions.com/>
[25] Tiers topology generator, available from <http://www.isi.edu/nsnam/dist/topogen/>
[26] NLANR measurement and operations analysis team. <http://moat.nlanr.net>