On the Availability of Replicated Contents in the Web

F. Tenzakhti & M. Ould-Khoua
Department of Computer Science,
University of Glasgow,
Glasgow G12 8RZ,
United Kingdom

K. Day
Department of Computer Science,
Sultan Qaboos University,
Muscat,
Oman

Abstract. *This study considers the problem of locating proxies in the Web in order to maximize object availability. A read one/write-all protocol is used and a placement based on the Dynamic Progamming (DP) technique is presented. The study then derives the properties of the resulting replicated system and studies its availability as a function of the read write ratio for uniform client requests.*

Keywords: *Tree network, Dynamic programming, object availability, Web.*

***Received:** July 16, 2003* | ***Revised:** February 15, 2004* | ***Accepted:** March 11, 2004*

1. Introduction

Being able to continue to provide services despite failures is one of the main advantages that replicated systems claim over single copy systems. In fact, the use of replication is attractive due to the potential improvement in availability, fault-tolerance, and performance[1,2,4,5,6,9]. Modelling and evaluation of replicated systems in the Internet is an important step in the design process of such systems. This study considers the problem of locating replicas in the Web in order to maximize object availability assuming a read-one/write-all protocol, proposes a placement algorithm, derives some properties of the resulting replicated system, and analyses its availability as a function of the read write ratio for uniform client requests. In such a protocol, a read request from a given client is serviced by the first proxy met along the way from the client to the target server. A write request from a client is first applied to the first proxy met while traveling up the tree to the target server and then propagated to the rest of the proxies.

The approach we adopt uses the dynamic programming technique (D.P). It builds on the work done by Bo.Li et al.[8] where they address the optimal placement of proxies in a tree network to optimise the performance of the Web. The novelty of our solution however is that it studies the availability of the Web instead of its performance. It also takes into account the capacity constraint of the nodes witch they ignored in their work. Considering the capacity of the nodes is very important because the number of requests serviced by a node affects the quality of service (QoS) (e.g. response time) provided to the requesting clients.

The rest of the study is organised as follows. Section 2 presents some work related to availability of replicated systems. Section 3 briefly describes the system model. Section 4 presents a technique for finding the optimal placement of a given number of proxies in a tree network that maximizes object availability. Section 5 presents an algorithm for finding the required number and placement of proxies that provides maximum availability. Section 6 derives the properties of the proposed algorithm. Section 7 analyses the availability offered by the algorithm for uniform client requests with a varying probability of link failure. Finally, Section 8 concludes the study.

2. Related Work

Most existing research work on availability in replicated systems has been conducted in the context of distributed systems [2,12]. To the best of our knowledge, there has been hardly any study that has considered the Web.

The authors in [4] have discussed the problem of placing data replicas in a ring network to maximize data availability. They have proved that the equal spacing placement is optimal for read-dominant systems, while the grouping placement is optimal for write-dominant systems. The drawback of this work is that it assumes a ring network, which is not suitable for an Internet environment.

In [9], a similar study has been carried out where the topology of the network is a tree. The objective is to maximize the probabilities of successful read-only and write-only transactions. The drawback of this research study is that it provides some properties for optimal placement without giving any real implementation results. A more related work is found in [10] where the authors have defined the core of a tree as a path for which the sum of distances is minimized. They have then defined a generalisation of a core, which they call a $k-tree$ core of a tree as follows. Given a tree T and a parameter k, a $k-tree$ core is a subtree T' of T containing exactly k leaves that minimizes $d(T')=\sum_{v\in V(T)} d(v,T')$ where $d(v,T')$ is the distance from vertex v to a subtree T'. They have then proposed two algorithms that find a $k-tree$ core of a tree with n vertices in $O(kn)$ and $O(n\lg n)$. Although this study has used similar assumption as in our work, their definition of the nearest replicas is different from ours. In our approach, each request from a client has to climb up the tree to the nearest proxy in the direction of the target server whereas in their approach the client reads from the nearest replica in any direction.

Kumar and Segev [7] have formulated three models to study the tradeoffs between availability and communication cost. In each model, the objective is to find a voting assignment that minimises communication cost. The analysis of one model has revealed that the read-one-write-all algorithm is optimal when the majority of operations applied to the replicated object are read only operations. Analysis in the two other models has shown that when the fraction of writes is relatively small, a voting assignment that satisfies a need for higher resiliency necessitates significantly greater communication cost.

3. The System Model

We consider the Web as of a collection of sites interconnected by a communication network. Objects are replicated at a number of sites and are managed by a group of processes called replicas, executing at the replica sites. The network topology is represented by a graph *G = (V, E)* where $m = |V|$ is the number of nodes and *E* is the set of links. The nodes are routers or Web servers or combination of both (servers provide the information we are looking for). Links correspond to physical links. Given the stability of Internet routing [9], an object requested by a client c and located at server s travels through a path, $s\rightarrow r_1\rightarrow r_2\ldots\rightarrow r_n\rightarrow c$, called preference path and denoted $\pi(s,c)$. This path consists of a sequence of nodes with the corresponding routers. Routes from s to the various clients form a routing tree along which requests are propagated. Consequently, for each Web server s, a spanning tree T_s rooted at s could be constructed to depict the routing tree, and the entire Web could be represented as a collection of such spanning trees, each routed at a given Web server (Fig 1).

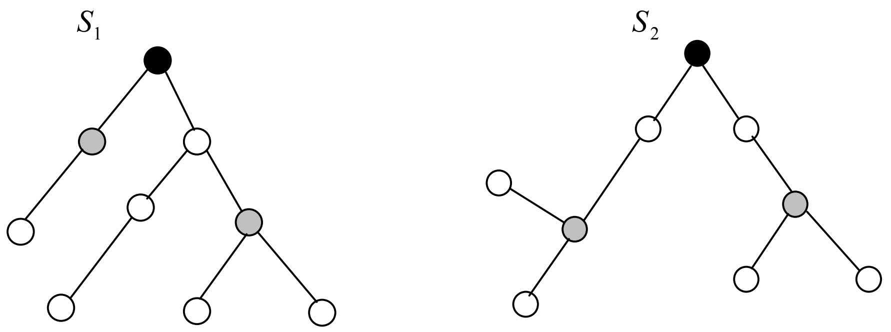

Figure 1: Two routing trees T_{s_1} and T_{s_2} for server s_1 and s_2 (dark) each with 2 proxies (gray)

Since an object from s to c passes by the nodes on the preference path $\pi(s,c)$, it would have been more secure if the request was serviced by one of the internal nodes in the path. In fact, the closer the data is on $\pi(s,c)$ to c, the greater the benefits.

3.1 Assumptions

Our study is based on the following assumptions, which have been widely used in existing studies [3,8]

- The Web is structured as a set of trees.
- An edge in a given tree fails with probability p, $0 \le p \le 1$. The failure of a node can be seen as a failure of an edge leading to that node.
- Each node v has a load λ_v (number of requests/sec.).
- Each node v has a capacity κ_v (number of requests/sec.) and the total loads of the clients serviced by v should not exceed κ_v.
- Objects accessed by clients are always available at proxies.

Table 1 provides the list of symbols used throughout this study.

As in [8], Consider a directed tree network T_s consisting of m nodes rooted at a server s and a residence set R of n proxy nodes replicating the target server s. Let T_s^R denote a replication tree of T_s which consists of the subtrees that spans all the proxies in T_s (Fig 2). The distance $d(i,j)$ between any two nodes $i, j \in T_s$ is the number of hops between i and j and the distance $d(i,T_j)$ is between the node i and the tree rooted at j, $d(i,T_j) = \min_{u \in T_j} d(i,u)$. A client at node i reads data located at the first proxy encountered while travelling to the targer server s which is the root of the tree T_s. We refer to this proxy as the optimal proxy and denote it by $p(i,s)$. Let ρ_i be the total number of read requests from node i in a fixed time interval θ, and similalry ω_i be the total number of writes from node i. The total number of read/write requests from the m nodes is $\tau = \sum_{i=1}^{m} (\rho_i + \omega_i)$. A read request from a client i is serviced by $p(i,s)$. A write request from i is first applied to $p(i,s)$ and then propagated to the rest of the proxies. Let r (resp w) be the rate of reads (resp writes) with respect to τ, $r = \sum_{i=1}^{m} \rho_i / \tau$, and $w = \sum_{i=1}^{m} \omega_i / \tau$. The probability of success of a read request from node i over the residence set R can be written as:

$$P_r(T_s, R, q) = q^{d(i,p(i,s))} \quad \text{where } q = 1 - p \ . \tag{1}$$

and the probability of success of a write request from node i is given by

$$P_w(T_s, R, q) = q^{d(i,p(i,s))+l-1} \ . \tag{2}$$

where l is the number of nodes of the tree T_s^R. Note that the write is performed at the first encountered proxy and then propagated to the n proxies over the tree T_s^R.

The availability, $A(T_s, R, q)$, of the whole system, which is the probabiblity of successful reads or writes over the residence set R can be written as:

$$\begin{aligned} A(T_s, R, q) &= \sum_{i=1}^{m} \frac{\rho_i}{\tau} q^{d(i,p(i,s))} + \sum_{i=1}^{m} \frac{\omega_i}{\tau} q^{d(i,p(i,s))+l-1} \\ &= \frac{1}{\tau} \sum_{i=1}^{m} [(\rho_i + \omega_i q^{l-1}) q^{d(i,p(i,s))}] \end{aligned} \tag{3}$$

Let node v whose optimal proxy is $u = p(v,s)$ imposes a load λ_v on u. Let $\vec{\kappa}$ be a vector representing the capacities of all the nodes in the tree and κ_V be the capacity of the node $v \in T_S$. Taking the capacity constraint into account the set P_V of nodes whose optimal proxy is v should not impose a load that is greater than the capacity κ_V of v. Consequently, if $P_v = \{x \in T_s : P(x,s) = v\}$, the inequality $\sum_{x \in P_v} \lambda_x \le \kappa_v$ must be satisfied. Assuming a fixed number of proxies $n \ge 1$, let us find the residence set R that maximizes the availability $A(T_s, R, q)$ over the tree T_s, taking into consideration the capacity constraints $\vec{\kappa}$ of the proxies. The problem thus reduces to maximising the following expression:

$$MaxA(T_s, q, \vec{\kappa}) = \max_{R \subset V} A(T_s, R, q) \text{ subject to}$$
$$\sum_{x \in P_V} \lambda_x \le \kappa_v \ , \forall v \in R. \tag{4}$$

4. Optimal Placement for a Fixed Number of Proxies

We use the dynamic programming technique to find the optimal number of proxies required in the system to maximize its availability. Given an integer $1 \le n \le m$, we are interested in finding a residence set R of size n that maximizes the above equation (4). Our approach is based upon the decomposition of the tree into three subtrees. A right subtree, whose availability is computed

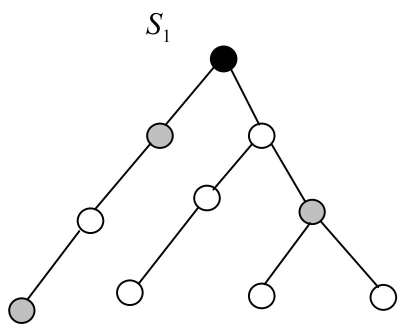

Fig. 4.2.a: The Tree T_s with 3 proxy servers (gray)

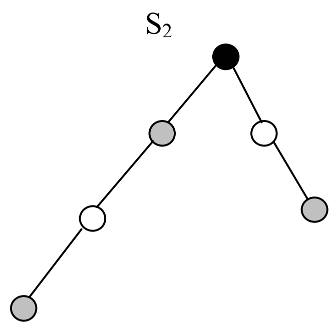

Fig. 4.2.b: The replication tree T_s^R

recursively, a middle subtree whose availabilitly has already been computed and a left subtree which is assumed to be empty of proxies, so that its availability can be easily computed.

Consider a tree T_s rooted at *s* with a set *V* of vertices. Assume that the children of each non-leaf vertex are ordered from left-to-right so that given any two siblings, u and v, we can determine whther u is to the left or right of v. Generally, given x and y in T_s, x is said to be the left of x if there exist u and v such that $x \in T_u$, $y \in T_v$ and u and v are siblings with u being to the left of v. For $v \in T_s$, let T_v be the subtree of T_s rooted at v. For $u \in T_v$, we can partition T_v into 3 subtrees:

1 Subtree $L_{v,u}$ containing all nodes to the left of u.
2 Subtree T_u containing all nodes in the subtree rooted at u.
3 Subtree $R_{v,u}$ containing the rest of the nodes.

Formally, we have

1 $L_{v,u} = \{x : x \in T_V : x \text{ is to the left of } u\}$
2 $T_u =$ subtree of T_V rooted at u
3 $R_{v,u} = \{x : x \in T_v, x \notin T_u \cup L_{u,v}\}$.

Figs. 3a, and 3b show a partitioning of the subtree T_V into three subtrees. The central approach of dnamic programming here is to divide the problem into small-scale sub problems. As a result, $R_{v,u}$ is further partitioned into smaller subtrees. Given the recursive nature of the solution, for all v in T_s, equation (4) applied to T_V yields:

$MaxA(T_v, q, \vec{\kappa}, n) = \max\limits_{R \subset V, |R| = n} A(T_v, R, q)$ subject to

$$\sum_{x \in P_v} \lambda_x \le \kappa_v \quad \forall v \in R. \tag{5}$$

where $MaxA(T_v, q, \vec{\kappa}, n)$ is the maximum availability obtained by placing n proxies in T_v given the load capacity vector $\vec{\kappa}$ of nodes in T_v .When $n = 1$, the only proxy is always placed at the root v. When $n > 1$, we can always find a node u, $u \in T_V$ $u \ne v$, which satisfies

1. A proxy is placed at node u;
2. No proxy is placed in $L_{v,u}$;
3. No proxy is placed along $\pi(u,v) - \{u,v\}$, which is the shortest path between nodes u and v not considering the nodes u and v.

Suppose that T_V is partitioned at the node u, and that n' proxies are placed in T_u, $1 \le n' \le n-1$, then $n - n'$ proxies are placed in $R_{v,u}$. We can therefore write:

$$MaxA(T_v, q, \vec{\kappa}, n) = \sigma_{L_{v,u}} p(L_{v,u}, q, \kappa_v) + \sigma_{T_u} MaxA(T_u, q, \vec{\kappa}, n') + \sigma_{R_{v,u}} MaxA(R_{v,u}, q, \vec{\kappa}', n - n') + wq^{d(u,v)} \tag{6}$$

where $p(L_{v,u}, q, \kappa_v)$ denotes the probability of accessing node v from all nodes in $L_{v,u}$,

and $\forall x \in R_{v,u}$ $\kappa'_x = \kappa_x$ and $\kappa'_v = \kappa_v - \sum\limits_{x \in L_{v,u}} \lambda_x$ (7)

and $\sigma_{T_v} = \sum\limits_{i \in T_v} (\rho_i + \omega_i) / \tau$ (8)

For all n proxies, we need to consider all possible partitioning points $u \in T_v$ and all possible n' values. Recursively, we put the proxies in T_u and $R_{v,u}$ the same way as in T_v. The dynamic programming approach can thus be formulated by the following equations :

$$MaxA(T_v.q,\vec{\kappa},n) = \begin{cases} \frac{1}{\tau}\sum_{i\in T_v}(\rho_i+\omega_i)q^{d(i,v)} & \mathit{if}\ n=1, and \sum_{x\in T_v}\lambda_x \le \kappa_v \\ \min_{u\in T_v,1\le n'\le n-1}(\sigma_{L_{v,u}}p(L_{v,u},q,\kappa_v) \\ +\sigma_{T_u}MaxA(T_u,q,n',\vec{\kappa})+\sigma_{R_{v,u}}MaxA(R_{v,u},q,n-n',\vec{\kappa}')+wq^{d(u,v)} & \mathit{if}\ n>1 and \sum_{x\in L_{v,u}}\lambda_x \le \kappa_v \\ \infty\ \mathit{Otherwise.} \end{cases} \quad (9)$$

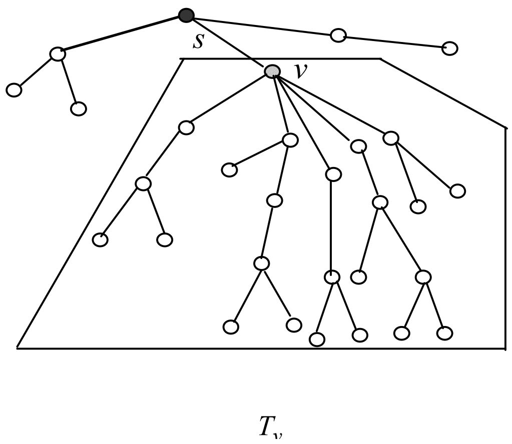

Fig.3a: The subtree T_v rooted at v .

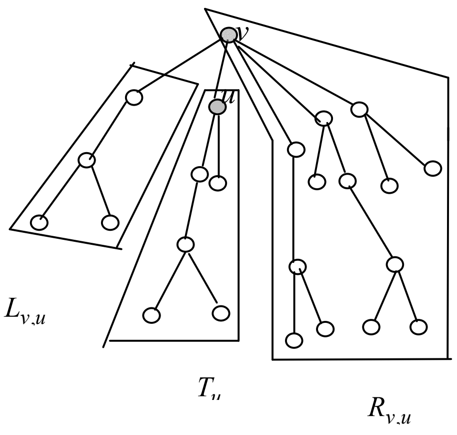

Fig. 3b Partitioning of T_v into $L_{v,u}$, T_u, $R_{v,u}$

In equation (9), the probability $p(L_{v,u},q,\kappa_v)$ is defined only if the total load of the nodes in $L_{v,u}$ is smaller than the capacity κ_v of the node v. $MaxA(T_u,q,\vec{\kappa},n')$ is recursively defined in T_v with the capacity constraints $\vec{\kappa}$ relative to the nodes $x \in T_u$. $R_{v,u}$ is further partitioned into $L_{v,u,u'}$, $T_{u'}$, and $R_{v,u'}$ around the node u' where a proxy is put. The capacity constraints of $R_{v,u}$ with respect to proxy v is the capacity vector $\vec{\kappa}'$ obtained by subtracting from κ_v the total load imposed on v by the nodes in $L_{v,u}$.

***Theorem 1**: Given a directed tree T_s of size m, the solution formulated by equation (9) for finding a residence set R of size n, $R \subset T_s$ that maximizes the availability of accessing objects in T_s has a complexity of $O(m^3n^2)$.*

Proof: As depicted in equation (9), $MaxA(T_u,q,\vec{\kappa},n)$ is defined for m^2 entries. This is because there are m ways of choosing u and at most m ways of choosing u'. Moreover, for any of the m ways of choosing u, there are $n-1$ different n' values. The same is true for the m ways of choosing u'. Since each entry requires $O(m)$ times for its evaluation, the complexity of the algorithm is thus

$O(((m)(m)(n-1)(n-1))(m)) = O(m^3n^2)$ ■

5. Optimal Placement for an Arbirtray Number of Proxies

In general, given a tree T_v whose root $v \in T_s$ is a proxy, we either place some proxies in T_v or none. Let $A_1(T_v,q,\vec{\kappa})$ be the availability obtained after placing no proxies in T_v except the root proxy, and $A_2(T_v,q,\vec{\kappa})$ obtained from the availability obtained after placing some proxies in T_v. The maximum availability is given by

$$MaxA(T_v, q, \vec{\kappa}) = \max\{A_1(T_v, q, \vec{\kappa}), A_2(T_v, q, \vec{\kappa})\}, \quad (10)$$

where

$$A_1(T_v, q, \vec{\kappa}) = \frac{1}{\tau}\sum_{i\in T_v}(\rho_i + \omega_i)q^{d(i,v)}$$

$$\text{subject to } \sum_{x\in T_v}\lambda_x \le \kappa_v \quad (11)$$

$$A_2(T_v, q, \vec{\kappa}) = \sigma_{L_{v,u}}\ p(L_{v,u}, q, \kappa_v) + \sigma_{T_u} MaxA(T_u, q, \vec{\kappa}) + \sigma_{R_{v,u}} MaxA(R_{v,u}, q, \vec{\kappa}') + wq^{d(u,v)} \quad (12)$$

$$p(L_{v,u}, q, \kappa_v) = \frac{1}{\sum_{i\in L_{v,u}}(\rho_i + \omega_i)}\sum_{i\in L_{v,u}}(\rho_i + \omega_i)q^{d(i,v)}$$

$$\text{subject to } \sum_{x\in T_v}\lambda_x \le \kappa_v \quad (13)$$

The placement in $R_{v,u}$ is carried out in a similar way to the placement in T_v yielding the following equations :

$$MaxA(T_v, q, \vec{\kappa}) = \max\{\frac{1}{\tau}\sum_{i\in T_v}(\rho_i + \omega_i)q^{d(i,v)}, \sigma_{L_{v,u}}\ p(L_{v,u}, q, \kappa_v) + \quad (14)$$
$$+ \sigma_{T_u} MaxA(T_u, q, \vec{\kappa}) + \sigma_{R_{v,u}} MaxA(R_{v,u}, q, \vec{\kappa}') + wq^{d(u,v)}\}$$

where $\forall v \in R_{v,u}, \kappa'_v = \kappa_v$ and $\kappa'_u = \kappa_u - \sum_{x\in L_{v,u}}\lambda_x$. (15)

The above equations compute the required number and placement of proxies to provide maximum availability.

Theorem 2 *The solution as formulated by equation (14) for finding the number of proxies required to maximize the availability for a tree T of size m can be computed in $O(m^3)$ time.*

Proof: $MaxA(T_s, q, \vec{\kappa})$ is defined for at most m^2 entries, and each entry requires $O(m)$ time for its evaluation. Therefore, the total order of the algorithm is $O(m(m^2)) = O(m^3)$.■

6. Proporties of The Proposed Solution

This section presents some properties of the replicated system.

6.1 Reads and Writes for a Single Copy

We consider the probability of successfully reading/writing one copy of an object located at the target server s. We assume that the requests are distrbuted randomly over the node of the tree T. Let $N_i(T_s)$ be the number of nodes in the tree T_s a distance i from s. The probability of a successful read request at node i is given by

$$P_r(T_s, R, q) = q^i. \quad (16)$$

Since the write is applied to the server only, the probability of a successful write is also

$$P_w(T_s, R, q) = q^i \quad (17)$$

Thus the availability of the system with a single copy placed at the target server s is given by

$$A(T_s, R, q) = \frac{(r+w)}{m}\sum_{i=1}^{m-1}N_i(T_s)q^i = \frac{1}{m}\sum_{i=1}^{m-1}N_i(T_s)q^i. \quad (18)$$

Theorem 3. *For a sufficiently small probability of failure, p, a necessary condition for maximizing the availability of the system is to place the target server s at the median of the tree T.*

Proof: When the probability of failure p is very close to 0, (q close to 1), $A(T_s, R, q)$ is maximized when $A'(T_s, R, q)$ is minimized. However,

$$A'(T_s, R, 1) = \frac{1}{m}\sum_{i=1}^{m-1}iN_i(T_s). \quad (19)$$

We wish to minimize the following expression:

$$f(1,s) = \sum_{i=1}^{m-1}iN(T_s) = 1N_1(T_s) + 2N_2(T_s) + 3N_3(T_s) + \ldots (20)$$

But $f(1,s) = \sum_{v\in T_s}d(v,s)$. Thus, the availability of the system is maximized when the target server s is the median of the tree T.

6.2 Reads for Multiple Copies

In this section, we assume that there are $n > 1$ copies, $c_1 \ldots c_n$, of a replicated object forming the residence set R and investigate the case of maximizing the probability, $P_r(T_s, q, R)$, of a succesful read from a node v over different replication sets R.

The placement of nodes in R induces a subtree T_s^R that spans all nodes in R and whose leaves are some subset of $c_1...c_n$. The root of T_s^R is the target server s and its nodes are $x_1, x_2, ... x_r$, $|T_s^R| = l \geq n$ (see Fig. 4). Let us refer to T_s^R as a "pure" proxy subtree when $l = n+1$, to mean that all the nodes T_s^R are proxies. Each node $i \in T_s^R$ is the root of an attached subtree T_i of nodes none of which except i belong to T_s^R. For every node $v \in T_s^R$, let $\alpha_v(q)$ be the probability that the node v successfully reads from the optimal proxy $p(v,s)$. In computing $P_r(T_s, q, R)$, we consider two cases: (1) the node v is in T_s^R, and (2) the node v is not in T_s^R.

1. If node v has a copy then $\alpha_v(q) = 1$. If v does not have a copy, then the read request from v has to travel in the direction of the root of T_s^R which is the target server s to find a proxy. This proxy could be s itself. The probability of a successful read is $\alpha_v(q) = q^{d(v,p(v,s))}$.
2. Node v is located in some subtree T_i for some $i \in T_s^R$. If $N_j(T_i)$ denotes the number of nodes belonging to subtree T_i that are a distance j from i, the probability of reaching one of the proxies is the probability of reaching i times $\alpha_i(q)$.

Considering both cases (1) and (2) we can thus write the following (20):

$$P_r(T_s, q, R) = \frac{1}{m}\sum_{i=1}^{l} \alpha_i(q) + \frac{1}{m}\sum_{i=1}^{l}(\alpha_i(q)\sum_{j=1}^{m-1} N_j(T_i)q^j)$$

Lemma 2. *When T_s^R is a pure proxy subtree then the probability of successful read is $\alpha_i'(1) = 0$, $\forall i \in T_s^R$.*

Proof. Since T_s^R is a pure proxy subtree then the probability of successful read $\alpha_i(q) = 1$, for any $i \in T_s^R$ thus $\alpha_i'(1) = 0$. ■

Theorem 4. *For n copies of a target server s grouped in a pure proxy subtree T_s^R, then for p sufficiently close to 0, a necessary condition for maximizing the probability of a successful read when n is close to l is to place the proxies such that $\min_{T_s^R \in T_s} \sum_{v \notin T_s^R} d(v, T_s^R)$.*

Proof. To maximize $P_r(T_s, q, R)$ over all T_s^R when p is sufficiently close to 0, we need to find the placement of T_s^R that minimizes $P'_r(T_s, 1, R)$. Since:

$$P_r(T_s, q, R) = (1/m)\sum_{i=1}^{l} \alpha_i(q) + (1/m)\sum_{i=1}^{l}(\alpha_i(q)\sum_{j=1}^{m-1} N_j(T_i)q^j)$$

we then have:

$$P_r'(T_s, q, R) = \frac{1}{m}\sum_{i=1}^{l} \alpha_i'(q) + \frac{1}{m}\sum_{i=1}^{l}[(\alpha_i'(q)\sum_{j=1}^{m-1} N_j(T_i)q^j) + \alpha_i(q)\sum_{j=1}^{m-1} jN_j(T_i)q^{j-1})] \quad (22)$$

Since $\alpha'_i(1) = 0 \, \forall i \in T_s^R$, we can write:

$$mP_r'(T_s, 1, R) = \sum_{i=1}^{l}\sum_{j=1}^{m-1} jN_j(T_i) = \sum_{v \notin T_s^R} d(v, T_s^R) \blacksquare \quad (23)$$

6.3 Reads and Writes for Multiple Copies

Assuming that reads and writes along with multiple copies, the availability of the system is given by :

$$A(T_s, q, R) = r[\frac{1}{m}\sum_{i=1}^{l} \alpha_i(q) + \frac{1}{m}\sum_{i=1}^{l}(\alpha_i(q)\sum_{j=1}^{m-1} N_i(T_i)q^j + w[\frac{1}{m}\sum_{i=1}^{l} \gamma_i(q) + \frac{1}{m}\sum_{i=1}^{l} \gamma_i(q)\sum_{j=1}^{m-1} N_i(T_i)q^j \quad (24)$$

When T_s^R is a pure proxy subtree, the availability of the system is maximized when the following expression is minimized:

$$A'(T_s, 1, R) = \frac{r}{m}\sum_{v \in T_s^R} d(v, T_s^R) + \frac{wn}{m}(m + \sum_{v \in T_s^R} d(v, T_s^R)) = \frac{r + wn}{m}\sum_{v \in T_s^R} d(v, T_s^R) + wn \quad (25)$$

Remark: When $r >> w$, which is a realistic assumption for the Internet, then even if n is large, $w \cdot n$ is small since w is small and the maximum availability is again obtained only if the nodes not belonging to T_s^R are not very distance from it; that if the quantity $\sum_{v \notin T_s^R} d(v, T_s^R)$ is minimized. If w is large however, the availability is maximized if l is small ($w(l-1)$ is small). In othe words, if the write rate is large, the size of the replication tree has to be small and the proxies should be close to each other so that the probability of successful writes is high and thus the availability is high.

7. Performance Analysis

This section analyses the availibity of the system as a function of the read write ratio and the probability of link failure. We consider the case of uniform request pattern and assume that all clients exihibt similar behaviour in that they issue the same of amount read and write requests. In this case, $\rho_i = \rho, \forall 0 \leq i \leq m$, and r now becomes $m\rho/\tau$, w becomes $m\omega/\tau$ and expression (3) becomes

$$A(T_s, R, q) = \frac{m\rho}{\tau}\sum_{i=1}^{m} q^{d(i,p(i,s))} + \frac{m\omega}{\tau}\sum_{i=1}^{m} q^{d(i,p(i,s))+l-1} \quad (26)$$

The above expression can be re-written as

$A(T_s, R, q) = rB + wC$, where: (27)

$$B = \sum_{i=1}^{m} q^{d(i,p(i,s))} \quad (28)$$

$$C = \sum_{i=1}^{m} q^{d(i,p(i,s))+l-1}. \quad (29)$$

B is the probability of success of a single read request over all the nodes and rB is the probability of success for the read rate r . C is the probability of success of a single write request over all nodes and wC is the probability of success for the write rate w. Since $w = 1 - r$, $(1 - r)C$ is the probability of success for write requests and thus the availability of the system is given by: $A(T_s, q, R) = rB + (1 - r)C = (B - C)r + C.$ (30)

This depicts a line on the (A, r) plane with $Y - intercept$ B= C , the probability of success of a write and slop $S = B - C$ which is the difference between the probabily of successes of a read and a write transactions over all nodes.

Remark: *The minimum slope is attained for one element replica set* $B = C$ *and* $S = 0$.

In order to analyse the availability of the system vith varying paramaters, we have plotted $A(T_s, q, R)$ against r , B, and C . In Fig. 4, the availability of the system is plotted against the read rate r , while varying the probability of a successful read, B , and maintaining a fixed value for the probability of a successful write C (e.g. C=0.9). The results reveal that when B is low (e.g. B=0.5), i.e. when the proxies are located farther away from the clients, the increase in the read rate decreases the availability of the system. However, as B increases (e.g. to B=0.9), the increase of the read rate r increases the availability of the system. Moreover, as r increases (i.e. the read activity increases), the availability of the system whose proxies are far away from the reading clients is higher compared to the system whose proxies are closer to the reading clients.

In Fig. 5, the opposite conclusions are reached. As the probability, C , of a succeffull write increases while maintaining a fixed value for B (e.g. B=0.9), the availablity increases with the increase in the read rate r . Furthermore, when the reading activity is low, the availability of the system varies with C ; being high when C is high. This is due to the fact that when C is high, a high writing activity yields a higher read-write success and thus a higher availability.

In Fig. 6 depicts availibity results when B and r have been varied while C is fixed at 0.9. The results reveal that firstly the availability of the system increases with the probability B , as is expected. Secondly, the availability is higher for lower read rates. This is explained by the fact that when B is small, the read requests are originating from far-away clients. As the number of these requests increases, the availability of the system decreases. This explains the reason why for all value of B , availability is higher when $r = 0.5$ than when $r = 0.9$. Finally, in Fig.7, B is fixed to 0.9 while C is varied from 0 to 1. For all reading rates, the availability increases with the increase in the rate. However, for low values of C , the availability is lower for lower reading rates (higher writing rates).

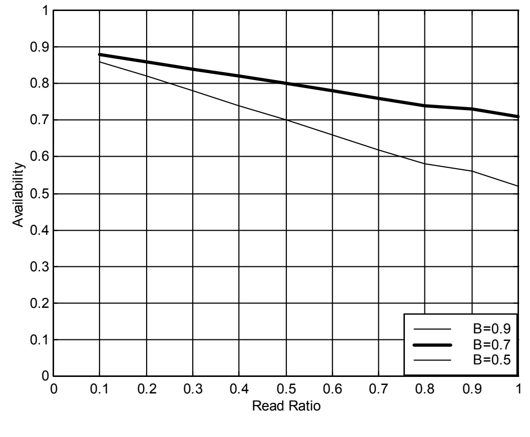

Fig. 4 Availability as a function of the read ratio. $B = 0.9, 0.7, 0.5$

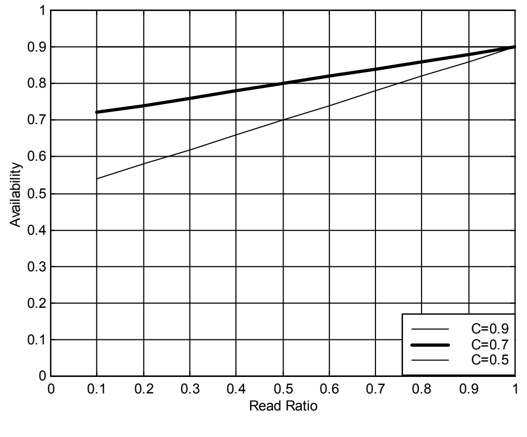

Fig. 5 Availability as a function of the read ratio $C = 0.9, 0.7, 0.5$

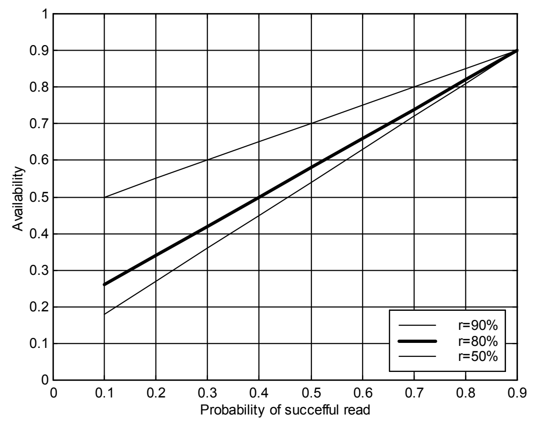

Fig. 6 availability as a function of the probability of successful reads $C = 0.9$

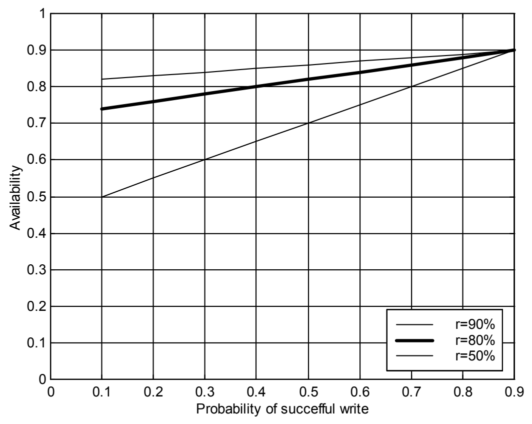

Fig. 7 availability as a function of the probability of successful writes $B = 0.9$

8. Conclusion

This paper has investigated using dynamic programming the problem of optimally placing a set of proxies in the Web in order to maximize system availability. The set of proxies have been modelled as a tree network. Firstly, a formulation of the problem for finding the optimal placement of n proxies to maximize availability in the tree network consiting of m nodes rooted at the target server s has been presented and an algorithm that runs in $O(m^3n^2)$ has also been suggested Secondly, a formulation of the problem for finding the otpimal required number and placement of proxies that maximizes availability has been proposed and algorithm that runs in $O(m^3)$ has also been proposed. Thirdly, the properties of the resulting proxy placement has been analysed and a study of the availability of the system has been conducted under both the uniform and non-uniform patterns for read/write request with varying read and write rates. The results of the analysis have revealed that in the case of uniform requests, (1) the availability of system increases as the read rate increases, (2) availability increases as the probability of success of read requests increases, (3) availability increases as the network size decreases, and (4) availability increases as the probability of success of writes increases.

References

1. L. Athanasio and I. Ahamed, "Static And Adaptive Data Replication Algorithms For Fast Information Access In Large Distributed Systems". *Int. Conf. Distributed Computing Systems*, Taipaei, Taiwan, pp. 7-15, Apr 2000.
2. B. Bhargava, A. Helal, and K. Friesen, "Analyzing availability of replicated database systems", *Int. Journal of Computer Simulation, vol. 1, no. 4,* pp 393-418,1991.
3. X.-D. Hu, X.-H. Jia, D.-Z.Du, D.-Y.Li, and H.-J. Huang, "Optimal Placement of Web Proxies for Replicated Web Servers in the Internet". *The computer Journal,* Vol. 44, No.5, pp-329-339,2001
4. X. D. Hu, X.H. Jia, D.Z, Du, D.Y Li, and H.J Huang, "Placement of data replicas for optimal data availability in ring networks"*, Journal of parallel and Distributed Computing,* vol. 61, no 1, pp.1412-1424, 2001.
5. R. Guerrouri, A. Shiper, "Software-Based Replication For Fault Tolerance", *IEEE Computer Society*, pp. 68-74. 1997.
6. A Helal. A. Heddaya, B. Bhargava, *Replication Techniques In Distributed Systems*, Kluwer Academic Publishers, 1996.
7. A. Kumar, and A. Segev, "Cost and Availability Tradeoffs in Replicated Data Concurrency Control", *ACM Transactions on Database Systems*, Vol. 18, No. 1102-131,1993.
8. B.Li, M.J. Golin, F. Italiano, X. Deng, K.Sohraby, "On The Optimal Placement Of Web Proxies In The Internet", *Proc. IEEE INFOCOM*, pp. 1282-1290, 1999.
9. V. Paxson, "End-to end routing behavior in the Internet". *IEEE/ACM Trans. Networking*, vol. 5, pp. 601-615, 1997.
10. S. Peng, A.B. Stephens, and Y. Yesha, "Algorithms For A Core And K-Tree Core Of A Tree", *Journal of Algorithms* , vol. 15, no 1, pp. 143-159, 1993.

11. L. Raab, "Bounds on the effects of replication on availability, *Proc. second IEEE Workshop Management of Data*, pp 28-40, Nov 1992.
12. A.B. Stephens, Y. Yesha and K. Humenik, Optimal Allocation for Partially Replicated Database Systems On Tree-Based Networks", *Appl. Math. Letters*, vol. 8, no.1, pp. 71-76, 1995.
13. F. Tenzakhti, K. Day, M. Ould-Khaoua, "Optimal Placement of Proxies in the Web While Maintaining Good QoS and Server Capacity Constraint", *The international Arab Conference On information Technology, ACIT2002,* pp 672-679, 2002.

Fathi Tenzakhti received the BSc degree in Computing and Information Sciences from Oklahoma State University (USA) in 1984 and the MSc in Computer Science from the Courant Institute of Mathematical Sciences at New York University (USA) 1987. He is currently working towards his PhD in the Department of Computing Sciences at the University of Glasgow in the area of distributed computing. His research interest are design, performance analysis and reliability study of replication algorithms in Web.

Khaled Day received an undergraduate degree in computer science from the University of Tunis in 1986 and the M.Sc. and Ph.D. degrees from the University of Minnesota (USA) in 1989 and 1992 respectively. Dr. Day is currently an Associate Professor at the Department of Computer Science of Sultan Qaboos University in Oman. His areas of interest include interconnection networks, parallel algorithms, distributed computing and cluster computing. He is a senior member of the IEEE.

Mohamed Ould-Khaoua received his B.Sc. degree from the University of Algiers, Algeria, in 1986, and the M.App.Sci. and Ph.D. degrees in Computer Science from the University of Glasgow, U.K., in 1990 and 1994, respectively. He is currently a Lecturer in the Department of Computing Science at the University of Glasgow, U.K. He is an Associate Editor for the International Journal of Computers & Applications and International Journal of High Performance Computing & Networking. He is the Guest Editor of five special issues on systems performance evaluation in the Journal of Computation & Concurrency: Practice & Experience, IEE-Proceedings-Computers & Digital Techniques, Future Generation Computing Systems, Journal of Parallel and Distributed Computing, and Performance Evaluation. He is the founding co-chair of the international workshop series on performance modeling, evaluation, and optimization of parallel and distributed systems (PMEO-PDS'02, PMEOPDS' 03, and PMEO-PDS'04). He has served on program committees of a number of well-known international conferences, including MASCOTS, SPECTS, and IC3N. Dr. Ould-Khaoua's current research interests are performance modelling of parallel and distributed systems, and wired/wireless networks. Dr. Ould-Khaoua is a member of the IEEE Computer Society.

Table 1. Notation

Symbol	*Meaning*
$A(T_s, R, q)$	availability of the system
$d(i, j)$	distance between i and j
E	set of Edges
l	size of the replication tree T_s^R
m	size of the tree
n	number of proxies
$N_j(T_i)$	number of nodes in T_i a distance j from i
$p(i,s)$	optimal Proxy for client i
p	probability of a link failure
$P_r(T_s, R, q)$	probability of success of a read transaction
$P_w(T_s, R, q)$	probability of success of a write request
q	probability of success
r	read rate
T_s^R	replication Tree for the target server s
T_i	tree rooted at i
V	set of vertices
w	write rate
ρ_i	number of reads of client i
ω_i	number of writes by client i
τ	total number of requests
λ_v	load imposed by node v
κ_v	capacity of a node v
γ_v (q)	probability of successfully writing from v to $p(v,s)$ and propagating it to the proxies
$\alpha_v(q)$	probability that v reads successfully from $p(v,s)$