An Arbitrary Tree-Structured Replica Control Protocol

Jean Paul Bahsoun, Robert Basmadjian
Université de Toulouse
Institut de Recherche en Informatique de Toulouse (I.R.I.T)
{bahsoun,basmadji}@irit.fr

Rachid Guerraoui
Ecole Polytechnique Fédérale de Lausanne
Distributed Programming Laboratory (LPD)
rachid.guerraoui@epfl.ch

Abstract

Traditional replication protocols that arrange logically the replicas into a tree structure have reasonable availability, low communication costs but induce high system load.

We propose in this paper the arbitrary protocol: a tree-based replica control protocol that can be configured based on the frequencies of read and write operations in order to provide lower system load than existing tree replication protocols, yet with comparable cost and availability. Our protocol enables the shifting from one configuration into another by just modifying the structure of the tree. There is no need to implement a new protocol whenever the frequencies of read and write operations change.

At the heart of our protocol lies the new idea of logical and physical levels in a tree. In short, read operations are carried out on any physical node of every physical level of the tree whereas the write operation is performed on all physical nodes of a single physical level of the tree.

We discuss optimal configurations, proving in particular a new lower bound, of independent interest, for the case of a binary tree.

1. Introduction

In large distributed systems, data is replicated in order to achieve fault-tolerance as well as improve system performance. However, underlying synchronization protocols, also known as replica control protocols, are required in order to maintain data consistency among the replicas. These protocols basically implement two operations: *read* and *write*. When one-copy equivalence is to be ensured, which is the case we consider in this paper, the write operation must store a new value of the data whereas the read operation must return the most recent value written.

Given the importance of the topic, several replication protocols have been described in the literature. They differ according to various parameters such as their communication costs, their ability to tolerate replica failures (also termed as their availability), as well as the load they impose on the system. For a system of *n* replicas, the well known *ReadOneWriteAll* (*ROWA*) [3] protocol has a read cost of *1* and a write cost of *n*. That is, a client needs only to contact a single replica to read a value, whereas it needs to contact all (*n*) replicas to write a value. Read operations are highly fault-tolerant and induce a load of $\frac{1}{n}$ whereas the availability of write operations is penalized due to the fact that the crash of a single replica prevents the write operations from terminating successfully; they induce a system load of *1*.

The *Majority Quorum* protocol [13] has both read and write communication costs of $\frac{n+1}{2}$ for an odd-sized number of replicas *n* and imposes a system load of at least *0.5*. It tolerates replica failures for read and write operations at the expense of increased read costs with respect to those of *ROWA*. Both *ROWA* and *Majority Quorum* protocols have a communication cost of $\mathcal{O}(n)$: the cost thus increases linearly with the number of replicas in the system.

By imposing a logical structure on the replicas of the system, it is possible to reduce the communication costs further. The $\sqrt{n}$ protocol (finite projective plane *FPP*) [9], the *Grid* protocol [4], the *Tree Quorum* protocol [2], the *HQC* protocol [8] and the *Paths system* [10] arrange logically the replicas of the system within a specific structure while, just like the *Majority Quorum* protocol, still making use of a quorum system. The load of these protocols was studied in [10] using linear programming. It was proven that the best optimal load of a quorum system of *n* replicas is $\frac{1}{\sqrt{n}}$ if the smallest quorum of the system is of size $\sqrt{n}$. It was also established that the load of the system becomes higher as much as the size of the smallest quorum of the system becomes less than $\sqrt{n}$.

The logical organization of *n* replicas of the system into certain structures such as a finite projective plane or a grid makes the smallest quorum become of size $\sqrt{n}$ resulting in the best optimal load of $\frac{1}{\sqrt{n}}$. The tree structures provide much smaller quorums of size $\log(n)$ but at the expense of very high system loads.

The motivation of this paper is to ask whether it is possible for a tree-based replica control protocol to induce a low system load with low communication costs and acceptable availability.

In a tree protocol, every node of the tree represents a replica of the system: the height of the tree is typically denoted by *h* whereas *n* denotes the number of replicas. One of the first tree-based protocols [1] has a read cost that varies from *1* to $(d+1)^h$ and a write cost of $\frac{(d+1)^{h+1}-1}{d}$ where each node of the tree has *2d+1* descendants (for a non negative value of *d*). In [7], the the idea of [1] was generalized to derive a protocol with a read cost that ranges from *1* to S^h whereas the write cost is of $\mathcal{O}(\log(n))$ where every node of the tree has *S* = 3 descendants. In [5], the proposed protocol requires a much smaller read costs than in [7] where such a protocol has a read cost that ranges from *1* to $S^{\frac{h}{2}}$ whereas the write operation has a cost of $\mathcal{O}(\log(n))$ where every node of the tree has *S* = 3 descendants. All these protocols have a read cost of *1* in the best case. However such a cost is achieved by accessing only the root of the tree: therefore the read operations of [1] and [7] having a cost of *1* induce a system load of *1* whereas those of [5] induce a system load of *0.5*. Also, all these protocols are vulnerable to the root crashing; in this case, no more write operations can be accomplished and these write operations impose a system load of *1* because the root is a member of every write quorum.

The drawback of not completing any write operation when the root of the tree crashes was solved in [2]. A read or write operation is performed by selecting a path that starts from the root of the tree and ends up at one of the leaves. If such a path cannot be constructed due to some failures then any inaccessible replica is replaced by paths that start from all of its descendants and end up at their corresponding leaves. This protocol has thus a read or write cost that varies from $\log n$ to $\frac{n+1}{2}$. For efficiency purposes, the authors considered the case of a binary tree. It was shown in [10] that the optimal system load of [2] is $\frac{2}{h+2}$ when performing read or write operations. However, this optimal load is achieved with a communication cost much higher than $\log n$. A communication cost (or a quorum size) of $\log n$ can be achieved by taking a path that starts from the root of the tree and ends up at one of the leaves. Such a quorum size of $\log n$ induces an optimal system load of *1* since every such quorum has the root of the tree as its member and any strategy of picking quorums of this size picks the root.

In *Hierarchical Quorum Consensus* (*HQC*) [8], a node does not necessarily correspond to a replica. The protocol is based on the *Quorum Consensus* approach and organizes the replicas of the system into a logical, multilevel hierarchy where only the leaf nodes of the tree represent replicas of the system. For efficiency purposes, the case of a ternary tree is considered and is proved that the size of the quorums of such a system is $n^{0.63}$ if the size of a quorum at each level is 2. It was shown in [10] that the optimal system load of *HQC* is $n^{-0.37}$which is higher than the best load of $\frac{1}{\sqrt{n}}$.

We propose in this paper a protocol, which we call the *arbitrary* protocol, and which somehow generalizes the protocol of [8]. As in the latter, we also organize logically the replicas into a tree structure and every node of this tree can be either *logical* or *physical:* a physical node corresponds to a replica of the system whereas a logical node does not. At the heart of our protocol lies the new notions of *physical level* which contains at least one physical node and *logical level* which has all of its nodes logical. Basically, the read operation is carried out on any single physical node at every physical level of the tree whereas the write operation is performed on all physical nodes of a single physical level of the tree. The structure of this tree can be configured based on the frequencies of read and write operations. If the write operations dominate in the system, then we add as much physical levels to the tree as possible. On the other hand, if the system is *mostly-read*, then all the replicas of the system are arranged into one physical level. We propose an algorithm that configures the tree structure by taking into account both read and write frequencies. We show that the write operations need only induce a system load of $\frac{1}{\sqrt{n}}$ with a communication costs of $\sqrt{n}$, which is lower than existing tree replication protocols. Yet we preserve comparable write availability. On the other hand, we show that the read operations need only induce a cost of $\sqrt{n}$ which is lower than the existing tree replication protocols with comparable system load and availability. We prove a new lower bound of the system load of the binary tree structure of [2] and show that the write operations of our protocol induce a load of $\frac{1}{log(n+1)}$ which is lower than the one proved in [10] of $\frac{2}{log(n+1)+1}$. Hence our approach provides a "spectrum" algorithm which can be tuned to suit the *read-write* ratio in the system.

The rest of the paper is organized as follows. We start by introducing few notations and give our representation of a tree structure using these notations. Then we detail how read and write operations of our protocol are performed and give for each operation its corresponding communication cost, availability and system load. We discuss the trade-offs in constructing the tree structure based on the frequencies of read and write operations.

2. Preliminaries

2.1. Definitions, notations and propositions

Most of the definitions, propositions and notations of this section are inspired by [10].

DEFINITION 2.1 *A Set system* $\boldsymbol{S}$ = { S_1, S_2, ..., S_m } *is a collection of subsets* $S_i \subseteq U$ *of a finite universe U. A*

Quorum system is a set system $\mathbf{S}$ *that has the intersection property:* $S \cap R \neq \emptyset$ *for all* $S, R \in \mathbf{S}$*. The sets of the system are called quorums.*

DEFINITION 2.2 *A coterie is a quorum system* $\mathbf{S}$ *that has the minimality property: there are no* $S, R \in \mathbf{S}$*;* $S \subset R$ *.*

DEFINITION 2.3 *A bi-coterie is a system* $\mathbf{S}$ *that has a separate set of read quorums* $\mathbf{R} = \{ R_1, R_2, \ldots, R_m \}$ *where* $R_i \subseteq U$ *and a separate set of write quorums* $\mathbf{W} = \{ W_1, W_2, \ldots, W_m \}$ *where* $W_i \subseteq U$ *, such that every read quorum intersects every write quorum, i.e.* $R \cap W \neq \emptyset \ \forall R \in \mathbf{R}$ *and* $\forall W \in \mathbf{W}$*.*

DEFINITION 2.4 *Consider a distributed system* $\mathbf{S} = \{ S_1, S_2, \ldots, S_m \}$ *over a finite universe U. We say that* $w \in [0,1]^m$ *is a strategy for* $\mathbf{S}$ *if it is a probability distribution over the subsets* $Sj \in \mathbf{S}$*, i.e.* $\sum_{j=1}^{m} w_j = 1$*.*

DEFINITION 2.5 *Let a strategy w be given for a system* $\mathbf{S} = \{ S_1, S_2, \ldots, S_m \}$ *over a finite universe U of size n. For a replica of the system* $i \in U$*, the load induced by w on i is* $l_w(i) = \sum_{i \in S_j} w_j$*. The load induced by a strategy w on a system* $\mathbf{S}$ *is:*

$$\mathcal{L}_w(\mathbf{S}) = \max \ l_w(i) \ \ \forall i \in U.$$

The system load on a system $\mathbf{S}$ *is given by:*

$$\mathcal{L}(\mathbf{S}) = \min \mathcal{L}_w(\mathbf{S}) \ \ \forall w \ ,$$

where the minimum is taken over all strategies w.

NOTATION 2.1 *Let n denote the number of replicas of the system and let h denote the height of the tree.*

NOTATION 2.2 *For a vector* $\mathbf{y} \in [0,1]^n$ *and a set* $S \subseteq U$*, let y(S) =* $\sum_{i \in S} y_i$*.*

NOTATION 2.3 *For a set of write quorums* $\mathbf{W}$*, let m(*$\mathbf{W}$*) denote the number of write quorums of* $\mathbf{W}$*. For a set of read quorums* $\mathbf{R}$*, let m(*$\mathbf{R}$*) denote the number of read quorums of* $\mathbf{R}$*.*

PROPOSITION 2.1 *Let a system* $\mathbf{S}$ *be given, and let w be a strategy for* $\mathbf{S}$ *with an induced load of* $\mathcal{L}_w(\mathbf{S}) = L$*. Then L is the optimal load iff there exists:* $\mathbf{y} \in [0,1]^n$ *such that y(U) = 1 and y(S)*$\geq L \ \forall S \in \mathbf{S}$ *.*

2.2. System model

A distributed system consists of a set of distinct sites that communicate by exchanging messages through communication links. A site consists of a processing unit and a storage device and has a unique *SID* whereas a communication link is a bidirectional communication medium between two sites. Sites can fail by stopping, as a result of the crash of a critical subsystem, or they can fail by performing arbitrary or malicious actions. We do not deal with such *Byzantine failures* [11] in this paper. We assume that sites fail independently with the same probability *q*. Moreover they are fail-stop and failures are transient and detectable. Links can fail either by not delivering messages at all to their desired destination, or by dropping or excessively delaying some of them. We also consider a special case of site and link failures that lead to the partitioning of the system where only sites in the same partition can communicate with each other. No assumption is made on the underlying topology of the network. We assume that sites of the system can be logically organized into a tree structure. Hence, we consider basic tree terminologies such as root, children, parent, leaves, height, node and level. Each client uses a centralized concurrency control scheme to synchronize accesses to the replicas. We assume that users interact with sites by means of transactions which are partially ordered set of read and write operations. Moreover, we assume that transactions are executed atomically, i.e. a transaction either commits or aborts at all participating sites. If a transaction contains write operations, a 2-phase-commit protocol at the end of the transaction is executed among all sites. Finally, we consider timestamps that consist of a version number and an *SID* which are used for read and write operations.

3. Our protocol

3.1. Notations and tree representation

Given a distributed system of *n* replicas, we organize them logically into a tree structure of height *h* where any non-leaf node of this tree can have any number of descendants. More precisely, let *S*(*i*,*k*) denote the i^{th} node of the k^{th} level of the tree where the orientation is taken from left to right and top to bottom respectively such that $i \in [1, m_k]$ and $k \in [0, h]$. Let m_k denote the total number of nodes at level *k* such that $m_k = \sum_{i=1}^{m_{k-1}} m(i, k-1) \ \forall k; k \in [1, h]$ and $m_0 = 1$. Let $m(i,k)$ denote the number of descendants of a non-leaf node *S*(*i*,*k*), where *i* and *k* have the same definitions as above, such that $m(i,h) = 0 \ \forall i; i \in [1, m_h]$.

A node of the tree is said to be *logical* if it does not represent a replica; otherwise the node is said

to be *physical*. Let $S_{log}(i,k)$ and $S_{phy}(i,k)$ denote a logical and a physical node of the tree respectively. Let $m_{log}(i,k)$ and $m_{phy}(i,k)$ denote respectively the number of logical and physical descendants of a non-leaf node $S(i,k)$ such that $m(i,k) = m_{log}(i,k) + m_{phy}(i,k)$, where $m(i,h) = m_{log}(i,h) = m_{phy}(i,h) = 0 \ \forall i; \ i \in [1, m_h]$. Then the total number of physical nodes at level k is denoted by: $m_{phy\,k} = \sum_{i=1}^{m_{k-1}} m_{phy}(i, k-1) \ \forall k; k \in [1, h]$ and $m_{phy\,0} = 1$ if the root of the tree is a physical node. On the other hand, the total number of logical nodes at level k is denoted by: $m_{log\,k} = \sum_{i=1}^{m_{k-1}} m_{log}\ (i, k-1) \ \forall k; k \in [1, h]$ and $m_{log\,0} = 1$ if the root of the tree is a logical node. Hence we have $m_k = m_{phy\,k} + m_{log\,k} \ \forall k; k \in [0, h]$ such that $m_0 = 1$ because the root of the tree is either a physical or a logical node.

A level of the tree is called *physical* if it contains at least one physical node. Let K_{phy} denote the set of all physical levels of the tree where $|K_{phy}|$ denotes the number of physical levels such that $1 \leq |K_{phy}| \leq h + 1$. Let $K_{phy}[u]$ indicate the level number of the physical level at index u of K_{phy} such that $u \in [0, |K_{phy}| - 1]$. We assume that the physical levels of K_{phy} are sorted in ascending order of their level numbers. On the other hand, we call a level of the tree *logical* if all of its nodes are logical, i.e. $m_k = m_{log\,k}$ and $m_{phy\,k} = \emptyset$. Let K_{log} denote the set of all logical levels of the tree and let $|K_{log}|$ denote the number of such logical levels where $0 \leq |K_{log}| \leq h$ and $|K_{log}| + |K_{phy}| = 1 + h$.

ASSUMPTION 3.1 *We assume that:*
$m_{phy\,0} < m_{phy\,1} \leq m_{phy\,2} \leq \ldots \ m_{phy\,h}$ *such that the total number of replicas of the system is given by:* $n = \sum_{k \in K_{phy}} m_{phy\,k}$

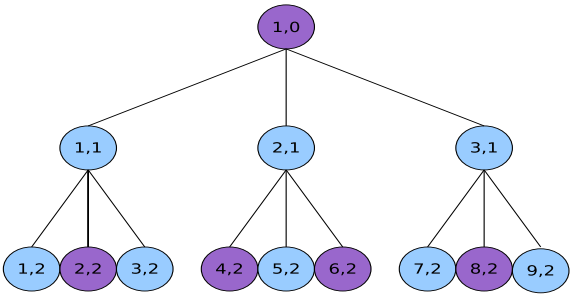

Figure 1. A tree whose purple circles represent logical nodes and the blue ones represent physical nodes. Such a tree has one logical level (0) and two physical levels (1 and 2).

3.2. The operations

Our system is a *bicoterie* where the set of read quorums $\boldsymbol{R}$ and the set of write quorums $\boldsymbol{W}$ are constructed based on definition 2.3. Furthermore, we assign separate strategies of picking read and write quorums using definition 2.4. More precisely, let w_{read} denote a strategy for picking read quorums of $\boldsymbol{R}$ such that $w_{read} = \sum_{j=1}^{m(\boldsymbol{R})} w_{read}\ j$ and let $w_{write} = \sum_{j=1}^{m(\boldsymbol{W})} w_{write}\ j$ denote a strategy for picking write quorums of $\boldsymbol{W}$ where $w_{read}\ j$, $w_{write}\ j$, $m(\boldsymbol{R})$, and $m(\boldsymbol{W})$ are defined in next sections. The availability computations are carried out by taking the assumption that every replica is *independently available* with a probability $p = 1 - q$ greater than $\frac{1}{2}$: it was proven in [12] that if the replicas are fail-prone, with an individual availability probability less than $\frac{1}{2}$, then the best strategy is not to replicate and to pick a *single centralized king*. In the rest of this section, we use d to denote the *minimal* number of physical nodes of the physical levels of the tree such that $d = \min \{m_{phy\,k} \ \forall k; k \in K_{phy}\}$, and use e to denote the *maximal* number of physical nodes of the physical levels of the tree such that $e = \max \{m_{phy\,k} \ \forall k; k \in K_{phy}\}$. Next, we demonstrate how we construct the read and write quorums and give the communication cost, availability and system load of read and write operations of our protocol.

3.2.1 The read operation

A read operation takes place by accessing all the members of a read quorum $R_j \in \boldsymbol{R}$ and retrieves the value of data whose timestamp has the highest version number and the lowest site identifier. A read quorum R_j is constructed by taking as its members any physical node of every physical level of the tree:
i.e. $R_j = \{S_{phy}(i,k) \ \forall k; k \in K_{phy}$ such that $\exists i; i \in [1, m_k]\ \}$.

FACT 3.2.1 *Let* $\boldsymbol{R} = \{ R_1, R_2, \ldots, R_j \}$ *be the set of read quorums such that every read quorum* R_j *is constructed in the same manner as explained in the previous paragraph. Then* $m(\boldsymbol{R}) = \prod_{k \in K_{phy}} m_{phy\,k}$.

In order to compute the load of the system induced by this read operation, we take a strategy w_{read} that picks each read quorum R_j with a probability $w_{read}\ j = \frac{1}{m(\boldsymbol{R})}$ where $j \in [1, m(\boldsymbol{R})]$.

The read operation of our protocol has a communication cost of $RD_{cost} = 1 + h - |K_{log}|$, an availability of $RD_{availability}\ (p) = \prod_{k \in K_{phy}} (1 - (1-p)^{m_{phy\,k}})$ and imposes

an optimal system load of $\mathcal{L}_{RD} = \frac{1}{d}$. It is important to note that as much as *d* increases, as much as the load imposed by the read operations on the system and communication cost diminish and the availability increases.

3.2.2 The write operation

A write operation, after obtaining the highest version number of data and incrementing it by one, accesses all the members of a write quorum $W_j \in \boldsymbol{W}$ in order to update their data with a new value and timestamp. A write quorum W_j is constructed by taking as its members all the physical nodes of any single physical level of the tree:
i.e. $W_j = \{S_{phy}(i,k)\ \exists k; k \in K_{phy}$ such that $\forall i; i \in [1, m_k]\ \}$.

FACT 3.2.2 *Let* $\boldsymbol{W} = \{ W_1, W_2,\ldots,W_j \}$ *be the set of write quorums such that every write quorum* W_j *is constructed in the same manner as explained in the previous paragraph. Then* $m(\boldsymbol{W}) = 1 + h - |K_{log}|$.

In order to compute the load of the system induced by this write operation, we take a strategy w_{write} that picks each write quorum of our system W_j with a probability $w_{write}\ j = \frac{1}{m(\boldsymbol{W})}$ where $j \in [1, m(\boldsymbol{W})]$.

The write operation of our protocol thus has a minimum cost of *d*, a maximum cost of *e* and an *average* cost of $WR_{cost} = \sum_{k \in K_{phy}} m_{phy\,k} * w_{write}\ j$. Hence such a strategy w_{write} of picking write quorums induces a communication cost of $\frac{n}{1+h-|K_{log}|}$. This operation has an availability of $WR_{availability}\ (p) = 1 - WR_{fail}\ (p)$ where $WR_{fail}\ (p) = \prod_{k \in K_{phy}} (1 - p^{m_{phy\,k}})$ and it imposes an optimal system load of $\mathcal{L}_{WR} = \frac{1}{1+h-|K_{log}|}$. It is important to note that as much as the *number of physical levels* of the tree increases, as much as the load imposed by the write operation on the system and communication cost diminish and the availability increases.

3.2.3 Discussion

The proofs of the *optimality* of system load induced by the operations of our protocol are provided in the *appendix* of this paper. The optimal system loads are computed by assuming that all replicas of the system are functioning properly. Therefore, the load of the system imposed by the operations becomes higher as the replicas of the system start to fail one after another. In order to compute the expected load knowing that replicas are available with a probability *p*, we use the following two equations:
EQUATION 3.2
$\mathbb{E}\mathcal{L}_{RD} = RD_{availability}(p) * (\mathcal{L}_{RD} - 1) + 1$
$\mathbb{E}\mathcal{L}_{WR} = WR_{availability}(p) * \mathcal{L}_{WR} + WR_{fail}(p) * 1.$

Therefore, we can notice from these two equations that as much as the availability of the operations is high, as much as the expected load is close to the computed system load induced by the operations. We call a system that has this characteristic *stable*.

Next, we demonstrate that our system is a *bicoterie* i.e. any read quorum has a *non empty intersection* with any write quorum. The read and write quorums of our protocol are of the following form:
Read Quorum = *any* replica of *every* physical level
Write Quorum = *all* replicas of *any* physical level

The proof is by induction on the number of physical levels of the tree:
Basis: Trivial for a tree of one physical level because all the replicas of the system are found at the same level.
Induction hypothesis: Assume that it holds for a tree of *h* physical levels: $|K_{phy}| = h$.
Induction Step: Consider a tree of $h + 1$ physical levels. Since every read quorum already had an intersection with every write quorum of *h* physical levels (*induction hypothesis* step) then it holds true because the fact of adding one new physical level does not prevent the read quorums to have a non empty intersection with any write quorum of *h* physical levels. On the other hand, since the read quorums contain a replica from the *new physical level* and the *new write quorum* contains all the replicas of this same new physical level, then any read quorum has a non-empty intersection with this write quorum. Hence, by induction, our protocol guarantees *non-empty intersection* of read and write quorums.

3.3. The tree organization and the trade-offs

Given a distributed system of *n* replicas, we organize them logically into a tree structure as explained in section 3.1 . However, this tree structure must be configured in such a way that it takes into account the *frequencies* of read and write operations of the system. For instance, as much as the number of physical levels of the tree increases, as much as the communication cost and the system load of the write operations diminish and the availability of the write operations becomes better. However, the fact of adding more physical levels to the tree results in increasing the communication cost and the system load of the read operations and in deteriorating their availability. Therefore, such a configuration is suitable for systems where write operations *dominates* the read ones. On the other hand, decreasing the number of physical levels of the tree to the *minimal* (only one physical level) diminishes the communication cost and the system load of the read operations and improves their availability. Therefore, such a configuration is appropriate for systems whose operations are *mostly-read* and such a configuration

behaves like *ROWA*. Finally, if both read and write operations happen in *proportional frequencies*, then the tree must be configured in such a way that both operations provide acceptable results. For a distributed system of *n* replicas where every replica is available independently with a probability *p* such that $n > 32$ and $p > 0.65$, the following organization of replicas always gives us satisfactory results for both read and write operations: $m_{log\,0} = 1$, $m_{phy\,1} = 4$, $m_{phy\,2} = 4$, $m_{phy\,3} = 4$, $m_{phy\,4} = 4$, $m_{phy\,5} = 4$, $m_{phy\,6} = 4$ and $m_{phy\,7} = 4$ whereas the remaining (n – 28) replicas of the system can be added to the succeeding physical levels of the tree in such a way that they obey the assumption 3.1. Next we give an algorithm to construct the arbitrary tree when $n > 64$:

Algorithm 1 An algorithm to construct the arbitrary tree

1- Construct the arbitrary tree of a *logical root node* by fixing $|K_{phy}| = \sqrt{n}$ and $h = |K_{phy}|$.
2- Arrange 4 replicas at the 1^{st} seven physical levels of the tree.
3- Arrange $\frac{n-28}{\sqrt{n}-7}$ replicas at every remaining physical level of the tree by obeying the assumption 3.1.

Therefore, when using *Algorithm* 1 to configure the tree, our protocol behaves in the following manner: the write operation has a *minimum* cost of 4, a *maximum* cost of $\frac{n-28}{\sqrt{n}-7}$ and an *average* cost of $\sqrt{n}$. It has a *failure probability* of $WR_{fail}\ (p) = (1-p^4)^7 * (1-p^{(\frac{n-28}{\sqrt{n}-7})})^{(\sqrt{n}-7)}$ and imposes an optimal system load of $\frac{1}{\sqrt{n}}$. On the other hand, the read operation has a *communication cost* of $\sqrt{n}$, an availability of $RD_{availability}(p) = (1-(1-p)^4)^7 * (1-(1-p)^{(\frac{n-28}{\sqrt{n}-7})})^x$ where $x = \sqrt{n} - 7$, and imposes an optimal system load of *0.25*. In order to study the *behavior* of the availability of the operations for very large number of replicas *n* such that $0.5 < p < 1$, we compute: $\lim_{n\to\infty} WR_{availability}(p) = 1 - \lim_{n\to\infty} WR_{fail}\ (p) = 1-(1-p^4)^7$ and $\lim_{n\to\infty} RD_{availability}(p) = (1-(1-p)^4)^7$. We can notice from these two equations that when $p > 0.8$, both operations have an availability of ≈ 1.

Finally, if we assume that our protocol is applied directly to a tree structure where every node of the tree has the same number of descendants and represents a replica of the system, then the write operations of our protocol impose an optimal load of $\frac{1}{log(n+1)}$ and these write operations are highly available (always greater than the availability probability *p*) and have an average cost of $\frac{n}{log(n+1)}$ whereas the read operations of our protocol impose the highest load of 1 to the system and these operations are poorly available (always less than the availability probability *p*) but has a communication cost of $\log(n+1)$. Furthermore,

m_k	$m_{phy\,k}$	$m_{log\,k}$
$m_0 = 1$	$m_{phy\,0} = 0$	$m_{log\,0} = 1$
$m_1 = 3$	$m_{phy\,1} = 3$	$m_{log\,1} = 0$
$m_2 = 9$	$m_{phy\,2} = 5$	$m_{log\,2} = 4$

Table 1. The total, physical and logical nodes of every level of the tree of Figure 1.

it was proven in [10] that the optimal system load imposed by read or write operations of [2] is $\frac{2}{log(n+1)+1}$ which is *higher* than the system load imposed by the write operation of our protocol. Therefore, we provide a new lower bound on the load of the tree structure of [2] imposed by write operations of our protocol.

3.4. An example

In this section, we give an example of a distributed system of 8 replicas which are organized logically into a tree structure as illustrated in Figure 1. We can notice that, such a tree has a height of 2, one logical level at zero, and two physical levels (1 and 2). Table 1 gives the corresponding numbers of total, physical and logical nodes of every level of the tree using the notations of section 3.1. Then we have:

- $n = m_{phy\,1} + m_{phy\,2} = 3 + 5 = 8$ such that this tree organization obeys to the assumption 3.1
- $K_{phy} = \{1,2\}$, $|K_{phy}| = 2$ and $K_{log} = \{0\}$, $|K_{log}| = 1$
- $m(\boldsymbol{R}) = m_{phy\,1} * m_{phy\,2} = 3*5 = 15$ and $m(\boldsymbol{W}) = 2$

Using the definitions of sections 3.2.1 and 3.2.2 we obtain:

- $RD_{cost} = 2$, $RD_{availability}\ (0.7) = 0.97$ and $\mathcal{L}_{RD} = \frac{1}{3}$
- $WR_{cost} = 4$, $WR_{availability}\ (0.7) = 0.45$ and $\mathcal{L}_{WR} = \frac{1}{2}$
- $\mathbb{E}\mathcal{L}_{RD} = 0.35$ and $\mathbb{E}\mathcal{L}_{WR} = 0.775$

In the rest of this paper, we represent such an arbitrary tree in the following manner: 1–3–5 where the numbers 3 and 5 denote the number of physical nodes at each physical level of the tree and "1" denotes a *logical root* node.

4. Configurations

In this section, we compare the communication costs as well as the system and expected system loads of read and write operations by setting up six different configurations. In the first one, which we call "*BINARY*", we take the case of a binary tree of [2] such that the computations of the load induced by read and write operations are based on the results of section 6.3 of [10]. The communication costs and

availability of the operations are based on the results of section 4 of [2] where the communication costs are computed by giving *f* a value of $\frac{2}{2+h}$ where *h* is the height of the binary tree and *f* indicates the fraction of quorums that include the root of a tree of level *l+1*. In the second case, we apply the read and write operations of our protocol directly to the binary tree structure of [2] without any modifications of this structure and we call such a configuration "*UNMODIFIED*". The third case is taken by creating a completely arbitrary tree of a *logical root* node by means of *Algorithm 1* and apply the read and write operations of our protocol where we call such a configuration "*ARBITRARY*". The "*HQC*" is studied by taking the tree structure of [8] where the computations of the load induced by read and write operations are based on the results of section 6.4 of [10] and the communication costs as well as availability of the operations are based on the results of sections 4 and 5 of [8]. In the fifth case, we configure the tree in such a way that the root is a *logical node* and all the replicas of the system are placed in *one and only one* physical level. By applying read and write operations of our protocol, it behaves like *ROWA* and we call such a configuration "*MOSTLY-READ*". Finally, the last configuration is determined by constructing the tree of a *logical root* node in such a way that for an odd-sized number of replicas, $|K_{phy}| = \frac{n-1}{2}$ and $h = |K_{phy}|$ such that every physical level consists of two replicas and we call such a configuration "*MOSTLY-WRITE*". It is important to note that, these last two configurations are special cases of the 3^{rd} configuration. The communication costs, availabilities and system loads of read and write operations of the configurations 2, 3, 5, and 6 are based on the definitions of sections 3.2.1 and 3.2.2.

The configurations are carried out by considering a distributed system composed of 15, 31, 63, 127 and 255 replicas. The followings are the representations of the arbitrary tree of the third configuration for the above-mentioned number of replicas respectively:1-3-3-3-3-3, 1-3-4-4-4-4-4-4-4, 1-4-4-4-4-4-5-5-10-10-13, 1-4-4-4-4-4-4-4-24-25-25-25 and 1-4-4-4-4-4-4-4-25-25-25-25-25-25-25-25-27.

4.1. The communication costs

Figure 2 illustrates the communication costs of read and write operations of our configurations. We can notice that, of the six configurations, "*MOSTLY-READ*" has the lowest communication cost of *1* for read operations and the worst cost of *n* for write operations. These costs are due to the way *n* replicas are arranged logically into *one and only one physical level*. For an odd-sized number of replicas *n*, "*MOSTLY-WRITE*" has the highest costs of $\frac{n-1}{2}$ for read operations and the lowest cost of *two* for write operations. This is because "*MOSTLY-WRITE*" maximizes the number of physical levels of the tree by adding at *each physical level two replicas*. However, these two configurations are suitable for specific environments where the read and write operations do not happen in proportional frequencies. Hence in the rest of this section, we compare the communication costs of read and write operations of the first four configurations: "*BINARY*", "*HQC*", "*UNMODIFIED*", and "*ARBITRARY*".

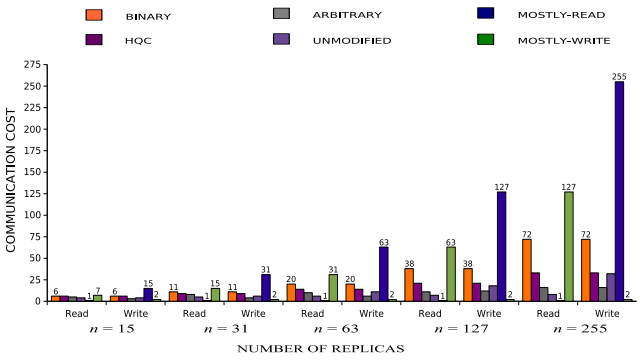

Figure 2. The communication costs of read and write operations of our configurations

Of these four configurations, "*BINARY*" has the highest costs of $\frac{2^h*(1+h)^h}{h*(2+h)^{h-1}} - \frac{2}{h}$ for both read and write operations where *h* denotes the height of the binary tree. On the other hand, "*ARBITRARY*" has the lowest communication costs for write operations whereas for read operations, it has lower costs than "*BINARY*" and "*HQC*", where the latter has a read and write costs of $n^{0.63}$, and has comparable costs with respect to "*UNMODIFIED*". Finally, this latter has the least communication costs of $\log(n+1)$ for read operations and has write costs of $\frac{n}{\log(n+1)}$ which is comparable with respect to "*ARBITRARY*" when $n < 200$ and comparable to those of "*HQC*" for $n > 200$.

4.2. The (expected) system loads

In this section, we present the (expected) system loads of read and write operations of our configurations. The expected system loads are computed using equations 3.2.

4.2.1 The read operation

The (expected) system loads of read operations are illustrated in Figure 3. We can notice that, of the six configurations, "*MOSTLY-READ*" has the lowest system load of $\frac{1}{n}$ and we can observe that such a system load is *stable* and diminishes as the number of replicas increases. On the other hand, "*MOSTLY-WRITE*" has a system load of $\frac{1}{2}$ for any number of replicas and such a system load is *instable* and

reaches easily to *1* due to the fact that at every physical level there are two replicas. In the rest of this section, we compare the system and expected system loads imposed by read operations of the first four configurations.

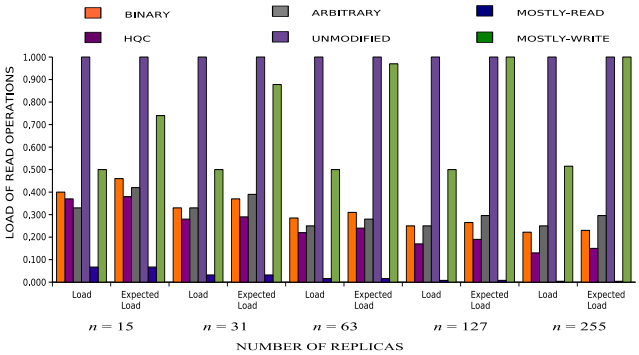

Figure 3. The (expected) system loads of read operations of our configurations

We can observe that "*UNMODIFIED*" has the highest system and expected system loads of *1* for any number of replicas among the six configurations. This is due to the fact that the root is a member of every possible read quorum and hence every read operation must access it. On the other hand, we can notice in Figure 3 that "*BINARY*", "*HQC*" and "*ARBITRARY*" have *quite stable* system loads which is due to the *high availability* of their read operations.

Among the first four configurations, "*HQC*" has the *least* system loads of $n^{-0.37}$ and the least expected system loads when $n > 15$. On the other hand, "*BINARY*" and "*ARBITRARY*" have quite *similar* (expected) system loads and that they are comparable to those of "*HQC*". Moreover, "*ARBITRARY*" has always a system load of $\frac{1}{4}$ when $n > 32$, whereas "*BINARY*" has always a system load of $\frac{2}{log(n+1)+1}$.

4.2.2 The write operation

The (expected) system loads of write operations are illustrated in Figure 4. We can notice that, among the six configurations, "*MOSTLY-READ*" has the highest (expected) system loads of *1* due to the fact that all *n* replicas of the system participate *in every write operation*. On the other hand, "*MOSTLY-WRITE*" has the least system load of $\frac{2}{n-1}$ for an *odd-sized number* of replicas *n* and such a system load is stable and diminishes as the number of replicas of the system increases. In the rest of this section, we compare the system and expected system loads imposed by write operations of the first four configurations .

We can observe that among the first four configurations, "*BINARY*" has the highest system load of $\frac{2}{log(n+1)+1}$ and the *highest* expected system load for any number of replicas. "*ARBITRARY*" has the *least* system load for any number of replicas. The load imposed by the write operations of

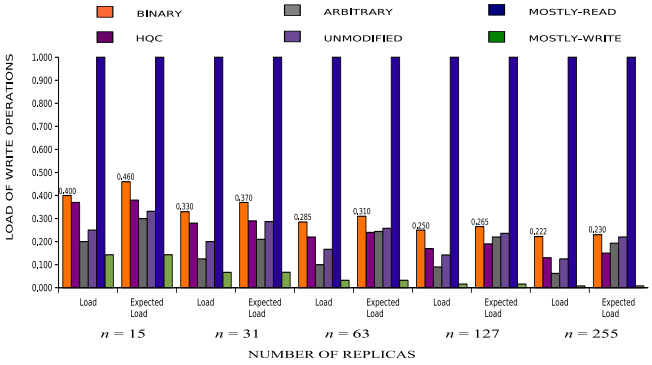

Figure 4. The (expected) system loads of write operations of our configurations

our protocol is always $\frac{1}{\sqrt{n}}$ whenever the tree is constructed using Algorithm 1. Furthermore, this configuration has the smallest expected system load for small number of replicas *n* and has comparable expected system loads with respect to "*HQC*" as the number of replicas *n* becomes larger due to the fact that the availability of the write operations of "*HQC*" is better than that of "*ARBITRARY*" when $p < 0.8$. Additionally, the write operations of "*HQC*" impose a system load of $n^{-0.37}$ and have the *best expected* system load for large number of replicas *n*. Finally, "*UNMODIFIED*" has the second lowest system load of $\frac{1}{log(n+1)}$ and has comparable expected system loads with respect to the configurations "*BINARY*", "*HQC*" and "*ARBITRARY*". However as we have seen in the previous section such a configuration has the *worst system load* for read operations. The expected system loads imposed by read and write operations of "*ARBITRARY*" become similar to the computed system loads, as the availability probability *p* of the replicas becomes higher than 0.8.

5. Conclusion

In a large replicated system, multiple copies of data must be kept synchronized by means of replica control protocols. Such protocols induce *trade-offs* between the communication costs and the system load. Whenever a low communication costs is desired for both read and write operations, a high load penalty must be paid by the system and vice versa. Tree-based replica control protocols can for instance provide low communication costs of $\mathcal{O}(log(n))$ for both read and write operations but at the expense of very high system loads.

In this paper, we proposed a new tree-based replication protocol: the arbitrary protocol. This can be configured appropriately based on the *frequencies* of read and write operations of the system. "*MOSTLY-READ*" is best suited for systems where *read* operations are carried out excessively, whereas "*MOSTLY-WRITE*" is appropriate for systems where *write* operations dominate. We identified a configuration which has the best combined read and write communication costs of $2 * \sqrt{n}$ and has the least system load of $\frac{1}{\sqrt{n}}$ imposed by write operations compared to the existing tree replica control protocols and its read operations always induce a system load of $\frac{1}{4}$. Our proposal enables the shifting from one configuration into another by just modifying the structure of the tree. There is no need to implement a new protocol whenever the frequencies of read and write operations of the system change.

Acknowledgement: We would like to thank the department of Mathematics of Paul Sabatier University and precisely Mr. Dominique Bakri and Miss Guitta Sabiini for their kind remarks on mathematical expressions.

References

[1] D. Agrawal and A. E. Abbadi. The tree quorum protocol: An efficient approach for managing replicated data. *Proceedings of the 16th VLDB Conference*, pages 243–254., 1990.

[2] D. Agrawal and A. E. Abbadi. An efficient and fault-tolerant solution for distributed mutual exclusion. *ACM Transactions on Computer Systems*, pages 1–20, 1991.

[3] P. Bernstein and N. Goodman. An algorithm for concurrency control and recovery in replicated distributed databases. *ACM Transactions on Distributed Systems*, 9 (4), 1984.

[4] S. Cheung, M. Ammar, and A. Ahamad. The grid protocol: A high performance scheme for maintaining replicated data. *Proc. IEEE Sixth Int'l Conf. on Data Engineering*, pages 438–445, 1990.

[5] S. Choi, H. Youn, and J. Choi. Symmetric tree replication protocol for efficient distributed storage system. *ICCS*, pages 474–484, 2003.

[6] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *J ACM 32*, pages 841–860, 1985.

[7] H. Koch. An efficient replication protocol exploiting logical tree structures. *The 23rd Annual International Symposium on Fault-Tolerant Computing*, pages 382–391, 1993.

[8] A. Kumar. Hierarchical quorum consensus: A new algorithm for managing replicated data. *IEEE Transactions on Computers*, pages 996–1004, 1991.

[9] M. Maekawa. A $\sqrt{n}$ algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems*, pages 145–159, 1985.

[10] M. Naor and A. Wool. The load, capacity, and availability of quorum systems. *SIAM J. Comput*, pages 423–447, 1998.

[11] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J ACM*, pages 228–234, 1979.

[12] D. Peleg and A. Wool. The availability of quorum systems. *Inform. and Comput*, pages 210–223, 1995.

[13] R. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems*, 4 (2):180–207, 1979.

6. Appendix: proofs of system load optimality

In this section, we provide the proofs of *optimal* system loads induced by read and write operations of our protocol.

6.1. The load of read operations

In this section, we prove that the *optimal* system load imposed by read operations of our protocol of *n* replicas arranged logically into a tree structure is: $\mathcal{L}_{RD} = \frac{1}{d}$ where $d = \min \{m_{phy\,k} \ \forall k; k \in K_{phy}\}$.

6.1.1 Proof of the upper bound: $\mathcal{L}_{RD} \leq \frac{1}{d}$

Let $\boldsymbol{R}$ = { R_1, R_2,...,R_j } be the set of read quorums such that every read quorum R_j is constructed in the same manner as explained in section 3.2.1. Then by Fact 3.2.1, we have $m(\boldsymbol{R}) = \prod_{k \in K_{phy}} m_{phy\,k}$.

Let $f(u) = \prod_{x=u}^{|K_{phy}|-1} m_{phy\ K_{phy}[x]} \times \prod_{y=0\ ;\ y \neq u}^{u-1} m_{phy\ K_{phy}[y]}$ and let $g(u) = m_{phy\ K_{phy}[u]}$. Then every replica $S_{phy}(i,K_{phy}[u])$ of the system is a member of m_q read quorums where m_q is given by: $m_q = \frac{f(u)}{g(u)} \quad \forall u; u \in [0, |K_{phy}| - 1]$ and $\forall i;\ i \in [1, m_{K_{phy}[u]}]$.

Let $w_{read} = \sum_{j=1}^{m(R)} w_{read}\ j$ be a strategy of picking read quorums $R_j \in \boldsymbol{R}$ as in definition 2.4 such that every $w_{read}\ j = \frac{1}{m(\boldsymbol{R})}\ \forall j; j \in [1, m(\boldsymbol{R})]$. It is clear that w_{read} is a strategy since $\sum_{j=1}^{m(R)} w_{read}\ j = 1$. By definition 2.5, the load induced by this strategy w_{read} on every replica of the system is: $lw_{read}\ (S_{phy}(i,K_{phy}[u])) = m_q \times w_{read}\ j = \frac{f(u)}{g(u) \times m(\boldsymbol{R})}$ $\forall u; u \in [0, |K_{phy}| - 1]$ and $\forall i;\ i \in [1, m_{K_{phy}[u]}]$.

However, $f(u) = \prod_{k \in K_{phy}} m_{phy\,k} \quad \forall u; u \in [0, |K_{phy}| - 1]$, then the load induced on every replica of the system is:

$$lw_{read}\ (S_{phy}(i,K_{phy}[u])) = \frac{1}{g(u)}$$

$\forall u; u \in [0, |K_{phy}| - 1]$ and $\forall i;\ i \in [1, m_{K_{phy}[u]}]$.
Hence by definition 2.5, the load induced by this strategy

w_{read} on the set of read quorums $\boldsymbol{R}$ is:

$$\mathcal{L}w_{read}(\boldsymbol{R}) = \max lw_{read}\ (S_{phy}(i,K_{phy}[u]))$$

$\forall u; u \in [0, |K_{phy}| - 1]$ and $\forall i;\ i \in [1, m_{K_{phy}[u]}]$.
Let $d = \min\{\ m_{phy\ K_{phy}[u]}\ \}\ \forall u;\ u \in [0, |K_{phy}| - 1]$ then $\mathcal{L}w_{read}(\boldsymbol{R}) = \frac{1}{d}$ and hence $\mathcal{L}_{RD} \leq \frac{1}{d}$. Keep in mind that $\mathcal{L}w_{read}(\boldsymbol{R})$ is the load induced by a strategy w_{read} on the set of read quorums $\boldsymbol{R}$. Next we have to prove that such a strategy is the optimal one that induces the optimal system load of $\mathcal{L}_{RD}$ when performing read operations of our protocol.

6.1.2 Proof of the lower bound: $\mathcal{L}_{RD} \geq \frac{1}{d}$

Given a tree of height *h*, let *t* denote the *first physical level* of this tree where the orientation is taken from top to bottom. Let $m_{phy\,t}$ denote the number of physical nodes at level *t*. Then by assumption 3.1, we have $m_{phy\,t} = d$ where *d* denotes the *minimal* number of physical nodes of the physical levels of the tree. Let us suppose that there exists a strategy w_{read} that induces a system load of $\mathcal{L}w_{read}(\boldsymbol{R}) = \frac{1}{d}$ when performing read operations of our protocol. Now we have to show that such a load is the optimal one. By using notation 2.2, then $\forall i; i \in [1, m_t]$ where m_t denotes the total number of nodes at level *t*, let $y_j = \frac{1}{d}$ for every replica $S_{phy}(i,t)$ and $y_j = 0$ for the replicas other than those at level *t*. Then clearly $y(U) = 1$ where *U* is the universe of the system and $y(R_j) = \frac{1}{d}$ for *every read quorum* $R_j \in \boldsymbol{R}$. Then by proposition 2.1 such a load is optimal and therefore $\mathcal{L}_{RD} \geq \frac{1}{d}$. Hence we deduce from the lower and upper bounds that $\mathcal{L}_{RD} = \frac{1}{d}$.

6.2. The load of write operations

In this section, we prove that the *optimal* system load imposed by write operations of our protocol on a distributed system of *n* replicas organized logically into a tree structure is: $\mathcal{L}_{WR} = \frac{1}{1+h-|K_{log}|}$ where *h* is the height and $|K_{log}|$ is the number of logical levels of the tree such that $0 \leq |K_{log}| \leq h$.

6.2.1 Proof of the upper bound: $\mathcal{L}_{WR} \leq \frac{1}{1+\boldsymbol{h}-|K_{log}|}$

Let $\boldsymbol{W}$ = { W_1, W_2,…,W_j } be the set of write quorums such that every write quorum W_j is constructed as explained in section 3.2.2. Then by Fact 3.2.2, we have $m(\boldsymbol{W}) = 1 + h - |K_{log}|$ where *h* is the height of the tree and $|K_{log}|$ is the number of logical levels. Furthermore, every replica $S_{phy}(i,K_{phy}[u])$ of the system is a member of one and only one write quorum W_j where $u \in [0, |K_{phy}| - 1]$ and $i \in [1, m_{K_{phy}[u]}]$.
Let $w_{write} = \sum_{j=1}^{m(W)} w_{write}\ j$ be a strategy of picking write quorums as in definition 2.4 such that every $w_{write}\ j = \frac{1}{m(W)}$. It is clear that w_{write} is a strategy since $\sum_{j=1}^{m(W)} w_{write}\ j = 1$. Then by definition 2.5, the load induced by this strategy w_{write} on every replica of the system is :

$$lw_{write}\ (S_{phy}(i,K_{phy}[u])) = \frac{1 * 1}{m(W)}$$

$\forall u; u \in [0, |K_{phy}| - 1]$ and $\forall i;\ i \in [1, m_{K_{phy}[u]}]$. Hence by definition 2.5, the load induced by this strategy w_{write} on the set of write quorums $\boldsymbol{W}$ is:

$$\mathcal{L}w_{write}(\boldsymbol{W}) = \max lw_{write}\ (S_{phy}(i,K_{phy}[u]))$$

$\forall u; u \in [0, |K_{phy}| - 1]$ and $\forall i;\ i \in [1, m_{K_{phy}[u]}]$. Then from the above equation we can say that: $\mathcal{L}w_{write}(\boldsymbol{W}) = \frac{1}{1+h-|K_{log}|}$.
Therefore, we can conclude from the proof that $\mathcal{L}_{WR} \leq \frac{1}{1+h-|K_{log}|}$.

6.2.2 Proof of the lower bound: $\mathcal{L}_{WR} \geq \frac{1}{1+\boldsymbol{h}-|K_{log}|}$

Given a tree of height *h* that has $|K_{log}|$ logical levels, then we have (1 + *h* – $|K_{log}|$) write quorums such that every replica *r* of the universe *U* belongs to *one and only one* write quorum. Let us suppose that there exists a strategy w_{write} that induces a system load of $\mathcal{L}w_{write}(\boldsymbol{W}) = \frac{1}{1+h-|K_{log}|}$ when performing write operations of our protocol. The next step is to prove that such a strategy is the *optimal* one that induces the optimal load. As we have mentioned above, the system consists of (1 + *h* – $|K_{log}|$) write quorums. By using notation 2.2, then $\forall u; u \in [0, |K_{phy}| - 1]$ we pick one replica $S_{phy}(i,K_{phy}[u])$ from every physical level of the tree and assign a value of $\frac{1}{1+h-|K_{log}|}$ to its y_j and a value of zero to the y_j' s of all other replicas $S_{phy}(i,K_{phy}[u])$ other than the picked one. Then clearly $y(U) = 1$ where *U* is the universe and $y(W_j) = \frac{1}{1+h-|K_{log}|}$ for every write quorum W_j. Then by proposition 2.1, such a load is optimal and therefore $\mathcal{L}_{WR} \geq \frac{1}{1+h-|K_{log}|}$.
Hence we deduce from the lower and upper bounds that $\mathcal{L}_{WR} = \frac{1}{1+h-|K_{log}|}$ where *h* is the height and $|K_{log}|$ is the number of logical levels of the tree such that $0 \leq |K_{log}| \leq h$.