Predicting Read- and Write-Operation Availabilities of Quorum Protocols based on Graph Properties

Robert Schadek, Oliver Kramer and Oliver Theel
Department of Computer Science,
Carl von Ossietzky University of Oldenburg, Germany

Keywords: Distributed Systems, Fault Tolerance, Data Replication, Quorum Protocols, Operation Availability Prediction, K-nearest Neighbors.

Abstract: Highly available services can be implemented by means of quorum protocols. Unfortunately, using real-world physical networks as underlying communication medium for quorum protocols turns out to be difficult, since efficient quorum protocols often depend on a particular graph structure imposed on the replicas managed by it. Mapping the replicas of the quorum protocol to the vertices of the real-world physical network usually decreases the availability of the operation provided by the quorum protocol. Therefore, finding mappings with little decrease in operation availability is the desired goal. The mapping with the smallest decrease in operation availability can be found by iterating all mappings. This approach has a runtime complexity of $O(N!)$ where N is the number of vertices in the graph structure. Finding the optimal mapping with this approach, therefore, quickly becomes unfeasible. We present, an approach to predict the operation availability of the best mapping based on properties like e. g. degree or betweenness centrality. This prediction can then be used to decide whether it is worth to execute the $O(N!)$ algorithm to find the best possible mapping. We test this new approach by cross-validating its predictions of the operation availability with the operation availability of the best mapping.

1 INTRODUCTION

Providing highly available access to a data object is a core problem in the field of computer science. Relying on a single replica of a data object greatly limits the availability of this data. This problem can be mitigated by creating multiple replicas of the same data. Using multiple replicas increases the availability of the data object as it can be accessed using different replicas. But replicating the data object also introduces the need for synchronization. Let the data object be replicated on five replicas, as shown in Figure 1. If the data object located on replica 0 is updated with a new value, then reading the data object from replica 4 does not yield the up-to-date value. Usually, this is not the intended behavior of a read operation. An additional problem is that two concurrent write operations can be executed on different replicas of the same data object at the same time. For example, value a is written to the data object on replica 2 and at the same time value b is written to the data object on replica 3. This raises the following question: Which value is the correct one? Both examples show that simply creating multiple replicas of a data object does not necessarily lead to the expected results. Usually, the goal is that all operations behave as they are expected to behave on a non-replicated data object. More formally, this *non-replicated behavior* can be achieved by a control protocol that guarantees *one-copy serializability* (1SR) (Bernstein et al., 1987). Many *quorum protocols* (QPs) implement such behavior. In general, QPs provide highly available access to data by means of replication and at the same time maintain 1SR. In most cases, QPs provide a read and a write operation to read and write data. QPs manage a set of replicas. These QPs use *read quorums* (RQs) and *write quorums* (WQs) to execute the desired operation. Quorums are specific subsets of replicas of the set of all replicas. Commonly, a read operation reads the data of all replicas of a RQ and identifies an up-to-date replica, for example, by means of version IDs. Write operations, on the other hand, use an atomic

Figure 1: Five replicas of a data object.

Schadek, R., Kramer, O. and Theel, O.
Predicting Read- and Write-Operation Availabilities of Quorum Protocols based on Graph Properties.
DOI: 10.5220/0006645705500558
In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018) - Volume 2*, pages 550-558
ISBN: 978-989-758-275-2

commit protocol, like the *Two-Phase Commit Protocol* (Bernstein et al., 1987), to write the new data to all replicas of a WQ and thereby providing a new highest version ID.

One possibility to achieve 1SR with a QP is to have all RQs intersect with all WQs, and all have all WQs intersect with all other WQs. Additionally, replicas contained in a WQ are locked for writing, and replicas contained in RQ are locked for reading. Any replica if locked can only exclusively be locked for reading or writing at any point in time. To execute a read (write) operation, all replicas in the RQ (WQ) have to be locked for the desired operation. As an example, consider a WQ consisting of three out-of five replicas of Figure 1. The write operation using this WQ writes the value c with a version ID 12 to three replicas $0, 1$, and, 2. Then, a read operation using a RQ, again with three out-of five replicas, is executed. This read operation reads a replica that hosts the last written data c with version ID 12, no matter which three replicas are chosen for the RQ. This QP is commonly known as the *Majority Consensus Protocol* (MCS) (Thomas, 1979), and is shown in more detail in Section 3.1.

In order for read (write) operations to work, the data of the operations need to be sent to and received by the replicas of the read (write) quorum used. Most QPs implicitly rely on a completely connected *graph structure* (GS) as a communication medium between its replicas (Schadek and Theel, 2017). We call this assumed GS a *logical network topology* (LNT). A *physical network topology* (PNT) is a GS that actually arranges and connects the replicas in the real-world. QPs have no influence on the PNT. In (Schadek and Theel, 2017), it is shown that the PNT used as a communication medium between the replicas, has to be considered in the cost and availability analyses of a QP to improve the accuracy of those analyses. Given a QP with N replicas and a PNT with N vertices, there are $N!$ possibilities to place these replicas on the N vertices of the GS of the PNT. One such assignment is called a *mapping*. Which mapping is chosen as the best one, can depend on different criteria. For example, the best mapping can be a mapping where the difference of the availability of the QP and the availability of the QP mapped to a PNT is the smallest. The selected criterion then has to be tested for all $N!$ mappings to find the best mapping. For a growing number of replicas, finding the best mapping becomes increasingly hard, due to the factorial nature of the problem.

The execution of the costly $O(N!)$ algorithm should be avoided, whenever the GS supposed to be used as a PNT is not well suited to be used as a PNT for the given QP.

We show how properties like e. g. *betweenness centrality* (BC) can be used to estimate the operation availability of a QP when mapped to a PNT. These properties are usually much easier to compute than the computational expansive $O(N!)$ mapping algorithm. Given the operation availability prediction based on these properties, the user then can decide whether it is worth finding the best mapping. The *K-nearest neighbor* (kNN) (Cover and Hart, 1967) approach is used for our predictions. We evaluate a number of properties and combinations of these properties for their prediction accuracy.

This paper is structured as follows. In Section 2, we present the system model used in this paper. In Section 3, related work and the evaluated properties are presented. The mapping approach is presented in Section 4. Section 5 discusses the use of kNN in the presented prediction approach. This section also includes an evaluation of the resulting approach. A conclusion and future work are given in Section 6.

2 SYSTEM MODEL

In order to analyze QPs, their characteristics, their use on PNTs, and the prediction capabilities of the properties, we first define a coherent model.

2.1 Graph Structure

A graph structure $\mathrm{GS} = (V, E)$ is a two-tuple of a set of vertices V and a set of edges E. Edges connect vertices. $V(\mathrm{GS})$ gives the set of vertices of a GS. $E(\mathrm{GS})$ gives the set of edges of a GS. A vertex $v \in V$ is a three tuple $v = (i, c_x, c_y)$, where $i \in \mathbb{N}$ is the ID of the vertex and c_x and c_y are the coordinates (i. e. its location of the vertex in the corresponding dimensions. The shorthand notation v_i gives a vertex with ID i. $N = |V(\mathrm{GS})|$ represents the number of vertices in a GS.

An *edge* $e_{i,j} \in E$ is defined as $e_{i,j} := (v_i, v_j)$, where $v_i, v_j \in V$.

A *path* $\langle v_0, v_1, \ldots, v_n \rangle$ between v_0 and v_n exists in GS, iff:

$$\forall i, 0 \leq i \leq n : \exists v_i \in V(\mathrm{GS}) \text{ and} \tag{1}$$

$$\forall i, 0 \leq i < n : \exists e = (v_i, v_{i+1}) \in E(\mathrm{GS}) \tag{2}$$

If a path exists, then the two vertices v_0 and v_n are called *connected.* $\mathbb{V}(\langle v_0, v_1, \ldots, v_n \rangle)$ denotes the set of vertices of a path such that:

$$\mathbb{V}(\langle v_0, v_1, \ldots, v_n \rangle) := \{v_0, v_1, \ldots, v_n\}. \tag{3}$$

$\mathbb{E}(\langle v_0, v_1, \ldots, v_n \rangle)$ denotes the set of the edges of a path such that:

$$\mathbb{E}(\langle v_0, v_1, \ldots, v_n \rangle) := \{e_{0,1} \ldots, e_{n-1,n}\}. \tag{4}$$

The shorthand notation "$\exists \langle v_0, v_1, \ldots, v_n \rangle \in GS$" can be used to state that the shown path exists in the GS.

Each vertex is assumed to be hosting exactly one replica of the replicated data object.

2.2 Graph Properties

The *betweenness centrality* (BC) property (Freeman, 1977) defines how often a vertex v occurs in all shortest paths of a GS, it is defined as:

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}. \tag{5}$$

Where $\sigma_{st}(v)$ gives the number of shortest paths between the vertices s and t in which vertex v is part of. σ_{st} represents the number of shortest paths between the vertices s and t. To use this vertex property in a way that describes the complete GS, we compute the minimum, the average, the mode, the median, and the maximum of the BC values of all vertices of the GS.

The second property we evaluate are distance properties between vertices. We call this property the *diameter* (Harary, 1969). Let $\varepsilon(v)$ be the longest shortest path of vertex v to any other vertex $v \in \mathbb{V}(GS)$. This is used to compute the minimum, the average, the mode, the median and the maximum distance based on all vertices $v \in V(GS)$.

The *degree* $deg(v)$ (Harary, 1969) describes how many edges a vertex v is connected to. Again, the minimum, the average, the mode, the median, and the maximum degree are considered.

Finally, the *connectivity* (Harary, 1969) is considered. Connectivity is the minimum number of vertices that need to be removed to disconnect one or more vertices from the rest of the vertices of the GS.

2.3 Consistency Criterion

In this paper, we discuss QPs that guarantee 1SR. Informally, 1SR states that read and write operations on replicated data objects have the same observable effect as operations on non-replicated data (Bernstein et al., 1987). QPs use quorums in the course of executing operations. Usually, QPs provide a read operation, using a RQ and a write operation, using on a WQ. Let Q be a QP providing read and a write operations that upholds the 1SR property. When 1) every RQ of Q intersects [1] with every WQ of Q, 2) all WQs of Q intersect with each other, and 3) replicas of Q can be locked exclusively for a read operation, or a write operation, then 1SR is guaranteed. Q upholds 1SR, since only a single write operation can write all replicas of its WQ, or one or more read operations can read the replicas of its RQ. This quorum intersection approach is used by many QPs to provide 1SR. This also holds for the QPs discussed in this paper.

2.4 Fault Model

Replicas are assumed to exhibit a fail-silent behavior. All failures are assumed to be independent of each other. The availability of a replica is described by p, where $0 \leq p \leq 1$. A p value of 1 means that the replica is available with a probability of 100% at an arbitrary point in time. A p value of 0 means that the replica is available with a probability of 0%. All replicas are assumed to have the same p value. Communication channels aka. edges are assumed to be always available. These simplifications gives way to a feasible analysis.

2.5 Read and Write Availability

The probability that a read or write operation is available for a given QP under the replica availability p is described by $a_r(p)$ and $a_w(p)$, respectively, where $0 \leq a_r(p), a_w(p) \leq 1$. The minimal average costs per read and write operation are given by $c_r(p)$ and $c_w(p)$. Let N be the total number of replicas.

$$\begin{aligned}
\text{RQS} = \{ & \{(q_1, \{sq_{1,1}, \ldots, sq_{1,m}\}), \\
& \ldots, (q_n, \{sq_{n,1}, \ldots, sq_{n,z}\})\} \quad | && (6)\\
& q_i \in \mathfrak{P}(\mathbb{V}(GS)) && (7)\\
& \wedge sq_{i,j} \in \mathfrak{P}(\mathbb{V}(GS)) && (8)\\
& \wedge isReadQuorum(q_i) && (9)\\
& \wedge (q_i, sq_{i,j}) : sq_{i,j} \supset q_i && (10)\\
& \wedge q_i, q_j : q_i \not\supseteq q_j && (11)\\
& \wedge (q_i, sq_{i,n}), (q_j, sq_{j,m}) : sq_{i,n} \neq sq_{j,m} && (12)\\
\}
\end{aligned}$$

$$\begin{aligned}
a_r(p) = & \sum_{\forall (q,sq) \in RQS} p^{|q|}(1-p)^{N-|q|} \\
& + \sum_{\forall (t \in sq \wedge (q,sq) \in RQS)} p^{|t|}(1-p)^{N-|t|} && (13)\\
ct_r(p) = & \sum_{\forall (q,sq) \in RQS} |q|(p^{|q|}(1-p)^{N-|q|}) \\
& + \sum_{\forall (t \in sq \wedge (q,sq) \in RQS)} |q|(p^{|t|}(1-p)^{N-|t|}) && (14)\\
c_r(p) = & \; ct_r(p)/a_r(p) && (15)
\end{aligned}$$

[1] Two quorums a and b intersect, if $a \cap b \neq \emptyset$.

For some QPs, there exists a closed formula to compute $a_r(p)$, $a_w(p)$, $c_r(p)$, and $c_w(p)$ respectively. In general, QPs allow to test whether a set of replicas is a RQ, or a WQ. Formula 9 shows how these tests can be used to compute the $a_r(p)$, and the $c_r(p)$. The formulas for $a_w(p)$, and $c_w(p)$ are analogous to $a_r(p)$, and $c_r(p)$. A *read quorum set* (RQS) is used to evaluate the $a_r(p)$, and the $c_r(p)$. It consists of a set of tuples that consists of a quorum q and a set of all other quorums that are supersets of q and are not present in any other such superset. Formulas 9 to 12 restrict the form of the set. The form of the sets is restricted in a way that no quorum appears more than once, as this would erroneously add to the availability mass of the set. $\mathfrak{P}(s)$ (Devlin, 1979) denotes the power set of all replicas of a QP. The value of $a_r(p)$ is then calculated by totaling the probability of the quorums being available as well as their supersets. $ct_r(p)$ serves as a temporary in the calculation of the $c_r(p)$. $ct_r(p)$ is calculated similar to the $a_r(p)$. For each q_i, and each $sq_{i,j}$ the availability is calculated. This availability is then multiplied with the number of replicas of q_i in each element of the RQS. The $ct_r(p)$ value is depended on the availability of the complete RQS. To remove this influence and thereby normalize the $a_r(p)$ over all p, $ct_r(p)$ is divided by $a_r(p)$. The result of the division is the $c_r(p)$. The number of replicas in q_i is also used for the supersets of q_i as it is assumed that QPs use smallest possible quorum. And the smallest possible quorum is represented by q_i. The calculation for $a_w(p)$ and $c_w(p)$ is only different in that they use the *write quorum set* (WQS) instead of the RQS. WQSs differs from RQSs in that its elements are WQs instead of RQs for the given QP.

3 DISCUSSION OF RELATED WORKS

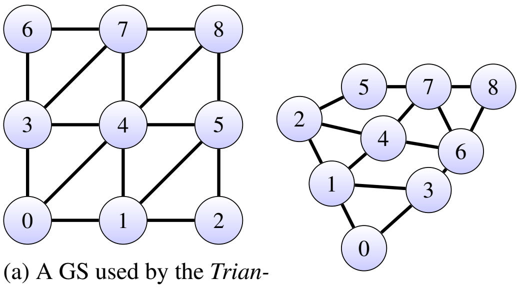
3.1 The Majority Consensus Protocol

The *Majority Consensus Protocol* (MCS) (Thomas, 1979) is a QP, that reads $\lceil N/2 \rceil$ replicas and writes $\lceil (N+1)/2 \rceil$ replicas, where N is the total number of replicas. The MCS guarantees 1SR. The read availability $a_r(p)$ of the MCS is

$$a_r(p) = \sum_{k=\lceil N/2 \rceil}^{N} \binom{N}{k} p^k (1-p)^{N-k} \tag{16}$$

and the write availability $a_w(p)$ is

$$a_w(p) = \sum_{k=\lceil (N+1)/2 \rceil}^{N} \binom{N}{k} p^k (1-p)^{N-k} \tag{17}$$



(a) A GS used by the *Triangular Lattice Protocol*.

(b) A random GS.

Figure 2.

(Koch, 1994). The MCS assumes that all vertices hosting the replicas are directly connected (Schadek and Theel, 2017). Therefore, the LNT implicitly used by the MCS is a *complete* GS (Chartrand et al., 2010). MCS works in the manner as described in Section 1.

3.2 The Triangular Lattice Protocol

The *Triangular Lattice Protocol* (TLP) (Wu and Belford, 1992) is an example of a QP using a LNT that is not a complete GS. Figure 2a shows an example of a GS used by the *Triangular Lattice Protocol* (TLP). The TLP is a very efficient QP only requiring $\sqrt{N}$ replicas to read and write in the best case, if the LNT used is a square. Every RQ consists of a complete vertical *or* a complete horizontal path through the GS. Every WQ consists of a complete vertical *and* a complete horizontal path through the GS. In Figure 2a, the diagonal path $\langle 0, 4, 8 \rangle$ connecting the replicas represents a minimal path that crosses the GS vertically as well as horizontally. This diagonal path, since it is very short, can therefore be used as a very efficient WQ. Due to the layout of the GS, vertical and horizontal paths always intersect in at least one replica. This way, it guaranties 1SR in the previously described way. As quorums for the TLP are created by finding paths through a GS, no simple closed formula exists yet that calculates the read and write availability. Therefore, we use the formulas presented in Section 2.5 for this purpose.

4 THE MAPPING APPROACH

A mapping is an injection from one GS to another GS. This requires that the number of replicas in the codomain structure is at least equal to the number of the replicas of the original structure. Formally, a mapping $M(\mathrm{GS}, \mathrm{GS}')$ from GS to GS$'$ is defined as:

$$M(\mathrm{GS},\mathrm{GS}') = \{(v_1, v_1'), \ldots, (v_n, v_n')\} \quad (18)$$

$$\forall (v, v') \in M : v \in V, v' \in V' \quad (19)$$

$$\forall (v, v'), (v, v'') \in M : v' = v'' \quad (20)$$

$$\forall (v', v), (v'', v) \in M : v' = v'' \quad (21)$$

(Schadek and Theel, 2017). When a mapping has been defined, the QP works on its LNT. For every replica selected to participate in a quorum by the QP, the mapping selects the mapped replica of the PNT. After the QP has selected all necessary replicas to construct a quorum based on its LNT, the mapping is used to tests whether the replicas are connected in the PNT. As mappings are usually not between homomorphic GSs, in the general case, it can be assumed that the replicas of the quorum are not directly connected with one another in the PNT. Consequently, a mapping has to add additional vertices to reestablish the connectedness and therefore the communication between the replicas of the quorum. The availability analysis of a mapped QP has to consider the additional replicas required on the PNT level. It is therefore a desired goal, to obtain mappings that require only few additional replicas on the PNT level in relation to the LNT level, as thus the expected availability characteristics of the QP on the LNT level can be matched the closest. For that reason, the important question is how to find the best mapping.

To find the best mapping, we need to compare mappings. In order to compare mappings, we have to define a comparison criterion. Any number of criteria can be selected depending on the intended use of the QP. Possible criteria are the average write costs, maximal read costs, average write costs, write availability, etc.

In this paper, we use the *average read and write availability value* (ARW) as the comparison criterion. The ARW approximates the weighted accumulation of the numerical integration of the read and write availability.

$$\mathrm{ARW} = (wor \cdot \sum_{p=0}^{100} a_w(p/100)) \cdot$$

$$((1 - wor) \cdot \sum_{p=0}^{100} a_r(p/100)) \quad (22)$$

$$wor \in [0, \ldots, 1] \quad (23)$$

The particular value of the *write over read* (*wor*) is a weighting factor between the read and write availability. A *wor* equal to 0.5 is used for the rest of the paper, to not favor any operation. The higher the ARW, the better is the mapping.

4.1 Optimal Mapping

In this section, we discuss the OPTIMALMAPPING algorithm. The algorithm finds the optimal mapping under the given ARW measurement criterion. “Optimal” in the scope of this paper means: the mapping where the ARW is the highest.

To get an intuition for the mapping approach, we give the following example. Let the TLP of Figure 2a be mapped to the PNT in Figure 2b with the mapping $M(\mathrm{GS},\mathrm{GS}') = \{(0,0), (1,1), (2,2), (3,3), (4,4), (5,5), (6,6), (7,7), (8,8)\}$. We assume that in the current state of the system, replicas $0,1,2,4,5,6,7,$ and 8 are available. Let the TLP have identified a WQ consisting of the replicas $0,4$ and 8. This is the currently availability WQ with the fewest replicas. None of these replicas are directly connected in the GS in Figure 2b under the current mapping. To reestablish communication between the replicas, we have to reconnect them with additional vertices. The fewer additional vertices the better. Two additional replicas are needed to reestablish communication between the elements of the WQ, e.g. replicas 3 and 6. The reconnected quorum consisting of five replicas is less likely to be available than the original quorum consisting of three replicas.

Given a PNT, a RQS, and a WQS, the procedure OPTIMALMAPPING, shown in Algorithm 3, finds the optimal mapping. The runtime complexity of the procedure OPTIMALMAPPING is $O(N!)$, where N is the number of vertices in the PNT. It is $O(N!)$, as there are $N!$ possible mappings for the N replicas to iterate. The algorithm does not require any knowledge of the QP used to generate the RQS and WQS, nor any knowledge of the LNT used by the QP. It only requires the RQS and the WQS created by the QP. This makes the algorithm applicable to a wide variety of QPs. The procedure APPLYMAPPING shown in Algorithm 1 is

Algorithm 1: Procedure APPLYMAPPING.

```
Input: quorums = (RQS,WQS)
       mapping = mapping to use
       GS = the graph structure to use
Result: a modified copy of quorums; where
        for each quorum in RQS and WQS
        the procedure FINDSMALLEST is
        called
a ←
    {FINDSMALLEST(MAP(q,mapping),GS)
    | q ∈ RQS}
b ←
    {FINDSMALLEST(MAP(q,mapping),GS)
    | q ∈ WQS}
return (a,b)
```

used for each mapping. The procedure APPLYMAPPINGS finds the smallest set of mapped vertices for each of the quorums in the RQSs and WQSs that reconnects the replicas of the quorums. APPLYMAPPING does this by calling the procedure FINDSMALLEST. The loop in line 3 in Algorithm 2 iterates the subsets in increasing order of the number of replicas contained in each subset. The result of APPLYMAPPINGS is then compared with the currently best known mapping. Depending on the comparison criterion, it is tested whether the currently tested mapping is the best mapping. If it is better, then the current mapping becomes the new best known mapping. After all mappings have been tested, the best known mapping is indeed the optimal mapping.

Algorithm 2: Procedure FINDSMALLEST.

```
Input:  GS = graph structure in which the
        verticesToConnect need to be
        connected in
        verticesToConnect = vertices for
        which a path needs to exists, so that
        they can communicate
Result: the smallest subset of $V(\mathrm{GS})$ for
        which the verticesToConnect are
        connected
1 $subsets \leftarrow \mathfrak{P}(\mathcal{V}(\mathrm{GS}))$
2 $smallest \leftarrow V(\mathrm{GS})$
3 forall $sbs \in subsets$ do
4     if $\forall i, j \in verticesToConnect$
      $|\exists\langle i, \dots, j\rangle \in GS \wedge |sbs| \leq |smallest|$
      then
5         $smallest \leftarrow sbs$
6     end
7 end
8 return smallest
```

5 THE MACHINE LEARNING APPROACH

In this paper, the aim of the machine learning approach is twofold. The first goal is to find out whether properties can be used to predict the read and write availability of QPs when mapped to a particular GS. If the first goal can be achieved, then the second goal is to identify properties or combinations of properties that yield the most accurate predictions.

The *K-nearest neighbor* (kNN) (Cover and Hart, 1967) (Bailey and Jain, 1978) approach is used to achieve these goals.

The basic idea of the approach is as follows. First, we compute RQS and WQS for a given QP with N replicas. Then, we generate a number of graphs with N vertices, that are not isomorphic to each other. For all these graphs, we compute the properties and standardize them (Mohamad and Usman, 2013). In the next step, we compute the optimal mappings of our tested QPs to all graphs. Then, we split the graphs into m equally sized parts in preparation for the *cross-validation* (CV) [2]. For all elements t in the powerset $\mathfrak{P}(T)$ where T is the set of all properties we execute the CV. During the CV, we select the kNNs based on t for the currently tested GS and mapping. Based on these k neighbors, we predict the read and write availability of the optimal mapping for the currently tested graph. Finally, we compare the prediction with the actual values by means of the *mean squared error* (MSE). Comparing different sets of properties based on the MSE allows us to identify properties that are well suited to estimate optimal mappings.

Algorithm 3: Procedure OPTIMALMAPPING.

```
Input:  $RWs, WQs$ = the RQS and WQS
        created by the unmapped QP
        GS = the graph structure, the QP
        should be mapped to
Result: the best mapping $i$ according to the
        user supplied mapping
        $optMapping \leftarrow$ empty tuple
forall $i \in \text{MAPPINGS}(GS)$ do
    tmp $\leftarrow$
      APPLYMAPPING((RQs,WQs), $i$, GS)
    if $tmp > optMapping$ then
        $optMapping \leftarrow$ tmp
    end
end
return optMapping
```

5.1 Test Data Generation

We generate two sets of random graphs. One set of graphs has eight vertices each, the other set of graphs has nine vertices each. Each set consists of 255 graphs. Having 255 graphs with nine vertices is currently the upper limit of what is possible to simulate in an acceptable time frame. No graph in a set is isomorphic to any other graph in its set. It exists a path between every vertex to any other vertex in the same graph. Each vertex is connected to one up to $N-1$ other vertex, where N is the number of vertices in the graph.

[2] In our case, m is equal to 5.

5.2 Implementation

The implementation is as follows. We begin by constructing RQS and WQS for the MCS for eight and nine vertices. We repeat this process for the TLP with a LNT being a 2×4, a 4×2, and a 3×3 grid. With the help of the OPTIMALMAPPING procedure, we determine the optimal mappings for the MCS and the TLP variants for all graphs in all groups.

Algorithm 5 (FINDBESTESTIMATOR) is the entry point into the finding of the best estimator or a combination thereof. The graph properties and their variants serve as estimators. The algorithm requires a set of graphs, the estimators to consider, the RQS, and the WQS created by the QP that is to be optimally mapped. In line 3 of Algorithm 5, the algorithm uses OPTIMALMAPPING to compute the optimal mapping of the given graph with the given parameter. The results of the OPTIMALMAPPING will be later used for the cross-validation. After the optimal mapping has been computed for all given graphs, the algorithm uses FINDBESTESTIMATORIMPL in Algorithm 5 to begin the comparison of the estimators. In line 3 in Algorithm 3 the set of graphs is partitioned into *k* equally sized subsets. This is done in preparation for the cross-validation. Starting in Algorithm 4, we begin the estimation process. $\mathfrak{P}(Es)$ yields the power set of the investigated estimators. In other words, we iterate all combinations of estimators starting on this line. The variable *tmpMSE* is used to accumulate the MSE produced by the iterations of the cross-validation starting of Algorithm 6. The *kNN* procedure is called for all graphs in *gss* and the prediction is stored in the variable *est*. The procedure COMPUTEMSE then computes the MSE between the estimation *est* and the actual optimal mapping *om*. After *m* executions of the cross-validation, it is checked in Algorithm 13 whether the current estimator has a smaller summarized MSE than the currently best one. If that is the case, then the current estimator is taken as the new best estimator. This part of the algorithm is simplified for readability reasons. Technically, actually two comparisons take place. These two comparisons compare the MSE of the read availability and the MSE of the write availability. This procedure continues until all elements of the power set have been tested. Finally, the best estimator and its MSE are returned. The choice of *k* for the kNN algorithm was empirically set to 7. We use five strategies to combine the seven estimations. These strategies are based on the minimum, the average, the median, the mode, and the maximum. For example, the *minimum* selects the minimum read and write availability prediction for all 101 *p* values from all seven neighbors.

Algorithm 4: Procedure FINDBESTESTIMATORIMPL.

```
Input: Gs = set of graphs to find the graph
       property-based estimator for
       Es = set of estimators
       Os = optimal mapping availability
       results for all the graphs in GS for a
       given QP
       m = number of subsets to use for the
       cross-validation
       k = number of neighbors to find
Result: set of estimators with the smallest
        MSE availability prediction and its
        MSE
1  $lowestMSE \leftarrow \{\}$
2  $bestEstimator \leftarrow \{\}$
3  $gss \leftarrow split(Gs,k)$
4  for $e \in \mathfrak{P}(Es)$ do
5      $tmpMSE \leftarrow \{\}$
6      for $i \in [0,m)$ do
7          for $g \in gss(i)$ do
8              $est \leftarrow kNN(k,g,e,gss,i)$
9              $om \leftarrow getOptimalMapping(g,Os)$
10             $tmpMSE = tmpMSE + computeMSE(est,om)$
11         end
12     end
13     if $tmpMSE < lowestMSE$ then
14         $lowestMSE \leftarrow tmpMSE$
15         $bestEstimator \leftarrow e$
16     end
17 end
18 return $(bestEstimator, lowestMSE)$
```

5.3 Evaluation

As mentioned in Section 2.2, the graph properties which we call base properties, *betweenness centrality* (BC), diameter, degree, and connectivity are evaluated as estimators. Except for the Connectivity property, all properties are vertex-based properties. But as we need properties for the whole graph, we compute the minimum, the average, the median, the mode, and the maximum of these four properties. It has also mentioned earlier that we are not only testing the individual properties, but also nearly all combinations of properties. Each combination of graph properties is only allowed to consist of different base properties. For example, a combination of 1) maximum Degree and 2) the median Degree has not been tested. Table 1 shows a selection of properties and combinations of properties that predicted the read and write availability of the optimal mapping best in at least one

Table 1: The graph properties and graph properties combinations used in the kNN predictions that lead to the best predictions in at least one instance.

Property \ ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
BetweennessAverage	x				x	x											x						x	
BetweennessMax				x					x												x			x
BetweennessMedian		x						x		x	x	x			x			x	x					
BetweennessMin																				x		x		
BetweennessMode			x				x							x		x								
Connectivity	x	x	x		x		x	x	x	x			x		x							x	x	
DegreeAverage	x		x								x	x	x	x					x					
DegreeMax								x														x		
DegreeMedian		x			x											x				x	x			
DegreeMin						x											x	x						
DegreeMode							x		x															
DiameterAverage	x	x	x				x			x	x		x	x		x		x		x	x			x
DiameterMax					x			x	x										x					
DiameterMedian																	x							

Table 2: MSE of the prediction of the read operation availability.

	Min		Avg		Median		Mode		Max	
	MSE	(ID)	MSE	(ID)	MSE	(ID)	MSE	(ID)	MSE	(ID)
MCS 8	**130.92**	**(6)**	0.00	(4)	0.00	(4)	3.65	(21)	149.21	(20)
MCS 9	84.16	(5)	0.00	(4)	0.00	(4)	2.98	(8)	175.70	(14)
TLP 4×2	0.22	(23)	0.00	(4)	0.00	(4)	0.22	(23)	0.22	(9)
TLP 2×4	0.67	(1)	0.00	(4)	0.00	(4)	0.67	(1)	0.45	(24)
TLP 3×3	8.34	(16)	0.00	(4)	0.00	(4)	1.07	(10)	66.48	(19)

Table 3: MSE of the prediction of the write operation availability.

	Min		Avg		Median		Mode		Max	
	MSE	(ID)	MSE	(ID)	MSE	(ID)	MSE	(ID)	MSE	(ID)
MCS 8	41.11	(3)	0.00	(4)	0.00	(4)	1.99	(18)	69.21	(9)
MCS 9	84.16	(5)	0.00	(4)	0.00	(4)	2.98	(8)	175.70	(14)
TLP 4×2	42.93	(11)	0.00	(4)	0.00	(4)	0.66	(12)	103.20	(17)
TLP 2×4	69.96	(7)	0.00	(4)	0.00	(4)	1.04	(13)	209.22	(15)
TLP 3×3	8.65	(2)	0.00	(4)	0.00	(4)	0.85	(10)	138.31	(22)

Algorithm 5: Procedure FINDBESTESTIMATOR.

```
Input: $Gs$ = The set of graphs to find the
       graph properties based estimator for
       $Es$ = The set of estimators
       $(RQs, WQs)$ = The RQS and WQS of
       the QP to find the best estimator for
$Os \leftarrow \{\}$
for $g \in Gs$ do
    $Os \leftarrow$
    $Os \cup OptimalMapping(RQs, WQs, g, r)$
end
$FindBestEstimatorImpl(Gs, Es, Os, 7, r)$
```

instance. Note the *ID* used to label the sets. Table 2 and Table 3 show the results of the predictions. Each of these tables show the MSE for the prediction of the read and write operation availability using the five combination functions for the kNN approach. In each entry, for instance “130.92 (6)” from Table 2, the first value represents the MSE and the second value given the ID of the estimator in Table 1. The set of properties identified by ID (6) consists of BetweennessAverage, and DegreeMin. The “x” marks the properties that belong to the property sets that is identified by their ID at the top of the table. The MSE is calculated over 101 p values that are scaled up from the range of $0 \leq p \leq 1.0$ to $0 \leq p \leq 100.0$. After the MSE evaluates the difference, it computes the power of that value. The power of these small values would be even smaller after taking them to the power of two. This goes against the intention of the MSE. The presentation of the MSE values was limited to two fractional digits, with the exception of the 0.0 value.

In all tables, the average and the median of the MSE is 0.0 for all QPs. In all cases, the estimator used is BetweennessMax.

The dominance of the BetweennessMax can be explained by looking into its meaning. Mappings are shortest paths through a graph. The BC property makes a statement about how often a particular vertex is part of all shortest paths through a graph. BetweennessMax expresses how often the replica, that is part of the most shortest paths in a graph, is part of a shortest path. Therefore, BetweennessMax basically states the BC value of the most important replica in the graph in regards of mapping quorums. As we are using graphs with the same number of vertices for each kNN iteration, BetweennessMax turns out to be a very good estimator for the quality of the mapping that we can expect from a graph. This is because graphs with the same BetweennessMax value have a very similar structure.

Overall, it can be said that the kNN methods in combination with graph properties predict the read and write operation availability of the optimal mapping very well. Especially, if the average or median is used in the predictions.

Since an MSE of 0.0 can not be improved, we refrained from testing methods like e. g support vector machines.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented an approach to predict the read and write availability of mappings based on graph properties. We have shown the high quality of these predictions based on five examples with 255 graphs. Additionally, we have demonstrated that betweenness centrality is a good property to use for predicting the read and write operation availability of mappings of Quorum Protocols. With the approach presented in this Paper, the reader has the opportunity to make an informed decision whether or not it is worth executing the computational expensive algorithm to determine the optimal mapping.

Going forward, we will test more graphs and graphs with more vertices. The next step would be to test with 12 vertices as this would allow to test the TLP on a 3×4 or 4×3 grids. Testing TLP on a 1×9, 1×10, or 1×11 grid is not useful, since degenerated grids will transform the analyzed TLP into the Read-One/Write-All protocol. In order to achieve this, first we have to improve the approach of finding an optimal mapping significantly. Currently, identifying an optimal mapping with nine vertices takes about seven hours. A graph with 12 vertices is 1320 times more complex and would therefore take about a year of computation time. Depending on the results obtained from these extended analyses, different prediction methods may be utilized.

Our second goal is to use the predictions of graphs with n vertices to give predictions of graphs with $n+m$ vertices, where $m > 0$. This approach could significantly reduce computation time.

REFERENCES

Bailey, T. and Jain, A. (1978). Note on distance-weighted k-nearest neighbor rules. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-8(4):311–313.

Bernstein, P. A., Hadzilacos, V., and Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Addison Wesley.

Chartrand, G., Lesniak, L., and Zhang, P. (2010). *Graphs & Digraphs, Fifth Edition*. Chapman & Hall/CRC, 5th edition.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Devlin, K. (1979). *Fundamentals of contemporary set theory*. Springer-Verlag, New York.

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41.

Harary, F. (1969). *Graph theory*. Addison-Wesley Publishing Co., Reading, Mass.-Menlo Park, Calif.-London.

Koch, H.-H. (1994). *Entwurf und Bewertung von Replikationsverfahren (in German)*. PhD thesis, Department of Computer Science, University of Darmstadt, Germany.

Mohamad, I. B. and Usman, D. (2013). Standardization and its effects on k-means clustering algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, 6.

Schadek, R. and Theel, O. E. (2017). Increasing the accuracy of cost and availability predictions of quorum protocols. In *22nd IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2017*.

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

Wu, C. and Belford, G. G. (1992). The triangular lattice protocol: A highly fault tolerant and highly efficient protocol for replicated data. In *Proceedings of the 11th Symposium on Reliable Distributed Systems (SRDS'92)*. IEEE Computer Society Press.