Quorum Placement in Networks to Minimize Access Delays

Anupam Gupta* anupamg@cs.cmu.edu
Bruce M. Maggs* bmm@cs.cmu.edu
Florian Oprea† foprea@ece.cmu.edu
Michael K. Reiter*† reiter@cmu.edu

ABSTRACT

A *quorum system* is a family of sets (themselves called *quorums*), each pair of which intersect. In many distributed algorithms, the basic unit accessed by a client is a quorum of nodes. Such algorithms are used for applications such as mutual exclusion, data replication, and dissemination of information. However, accessing spread-out quorums causes access *delays* that we would like to minimize. Furthermore, every member of the quorum incurs processing *load* to handle quorum accesses by clients.

In this paper we study the problem of placing quorums in a physical network so as to minimize the delay that clients incur by accessing quorums, and while respecting each physical node's capacity (in terms of the load of client requests it can handle). We provide approximation algorithms for this problem for two natural measures of delay (the *max-delay* and *total-delay*). All our algorithms ensure that each node's load is within a constant factor of its capacity, and minimize delay to within a constant factor of the optimal delay for all capacity-respecting solutions. We also provide better approximations for several well-known quorum systems.

Categories and Subject Descriptors: C.2.4 [Computer-Communication Networks]: Distributed Systems - *distributed applications*

General Terms: Algorithms, Performance, Theory.

Keywords: Quorum Systems, Location Problems, Approximation Algorithms, LP Rounding.

1. INTRODUCTION

Given a *universe* U of elements, a *quorum system* $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ on U is a family of subsets of U such that any two quorums Q_i and Q_j have a non-empty intersection. Quorum systems arise naturally in many algorithms in distributed systems for achieving mutual exclusion, consistent data replication, and dissemination of information (see, e.g., [5, 8, 15, 19]). In typical quorum-based algorithms, each client accesses the system by accessing all the elements in some quorum $Q_i \in \mathcal{Q}$. While the intersection property ensures that any Q_i would suffice, usually the client chooses Q_i from a probability distribution $p : \mathcal{Q} \to [0, 1]$ over $\mathcal{Q}$; this p is called the *access strategy* for the quorum system. The access strategy p is typically chosen to minimize the *load* of the most heavily loaded element $u \in U$, where the load on $u \in U$ is defined by $\mathsf{load}_p(u) = \sum_{Q_i \ni u} p(Q_i)$. Much research in the past has led to the development of very good quorum systems $\mathcal{Q}$ and associated access strategies p to minimize load [18].

1.1 Load versus Delay

Most of this prior research assumes an abstract setting which does not ascribe any costs or delays to quorum accesses or heterogeneous limits on the capabilities of different elements. However, in many network-based applications, one has to map the "logical" elements of the universe U on the "physical" nodes of a given network, which gives rise to many interesting issues. In practice, the physical nodes may have different capacities to handle quorum accesses, e.g., due to different computing capabilities. (For instance, one does not want a PDA on the network to be using all its computing resources to serve quorum accesses.) Hence, we want to ensure that physical nodes handle no more load than their "capacity".

Equally importantly, since the accesses of quorums by clients (which are themselves nodes in the network) have to be implemented by messages sent along the network, the performance of quorum-based systems now crucially depends on the *delays* introduced by these accesses. In fact, one would like the logical quorums $Q_i \in \mathcal{Q}$ to be mapped to closely clustered physical nodes in the network so that we do not incur large delays in trying to reach far-flung parts of the network. Of course, there is a natural tension between these two desires of low load and low delay: one can achieve an excellent clustering by mapping *all* the universe elements to a single physical node, but this would create a huge load on that node!

Only recently have these issues arising from mapping quorum systems to physical networks started to receive attention in the literature. (See Section 2 for a discussion of the prior work.) In this paper, we define and investigate some natural quorum placement problems. We present algorithms to map quorum systems onto physical networks so as to keep both access delays and loads on physical nodes in

*Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
†Electrical & Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA, USA

PODC'05, July 17–20, 2005, Las Vegas, Nevada, USA.

check. Loosely, our algorithms take as inputs:

- a quorum system $\mathcal{Q}$ defined on logical elements U, along with an access strategy p for $\mathcal{Q}$,[1] and
- a network $G = (V, E)$, where each physical node $v \in V$ prescribes an upper bound $\mathsf{cap}(v)$ on the quorum load it can be assigned.

The output is a mapping of the logical nodes U onto the physical nodes V that achieves *"small delay"*—i.e., quorums are placed close to clients—and such that each physical node has a *"small load"*—i.e., a load that does not exceed its capacity.

1.2 Results

More precisely, we are given an undirected network $G = (V, E)$, of size $|V| = n$, with each node having an associated capacity $\mathsf{cap}(v) \in \mathbb{R}^+$. There is a positive "length" $\mathsf{length}(e)$ for each edge $e \in E$, which induces a distance function $d : V \times V \to \mathbb{R}^+$ obtained by setting $d(v, v')$ to be the sum of lengths of the edges comprising the path from v to v' that minimizes this sum (i.e., the shortest path). We assume that the set of clients wanting to make quorum accesses is V.

Load. Given a quorum system $\mathcal{Q}$ over U and an access strategy p, this induces a load on each element $u \in U$, given by $\mathsf{load}(u) = \sum_{Q \in \mathcal{Q}: u \in Q} p(Q)$. Our goal is to determine a map $f : U \to V$ (which we call a *placement* of the quorum $\mathcal{Q}$ on the nodes of G) so that $\mathsf{load}_f(v)$ of any node $v \in V$ is at most the upper bound $\mathsf{cap}(v)$; we define $\mathsf{load}_f(v) = \sum_{u \in U: f(u) = v} \mathsf{load}(u)$.

Delay. We mainly deal with the following notion of delay, called the *average max-delay*. Given a placement f, if a client $v \in V$ accesses a quorum Q, then the time required to reach all elements of quorum Q is proportional to the *maximum* distance from v to any element of the quorum Q. Hence we model the *delay* as the distance of v to the farthest-away element of Q:

$$\delta_f(v, Q) = \max_{u \in Q} d(v, f(u)). \quad (1)$$

(We call this the *max-delay* access cost.) Then, the *expected delay* (under p) for v to access $\mathcal{Q}$ is

$$\Delta_f(v) = \sum_{Q \in \mathcal{Q}} p(Q)\, \delta_f(v, Q). \quad (2)$$

Note that if each client $v \in V$ makes quorum accesses at the same rate, then the average delay of the entire system will be $\mathsf{Avg}_{v \in V}[\Delta_f(v)] = \frac{1}{n} \sum_v \Delta_f(v)$. (For ease of exposition, we focus on this uniform-rate case; we can extend our results to more general cases when different clients make quorum accesses at different rates.)

We are finally in a position to define our main problem formally:

PROBLEM 1.1. **(Quorum Placement Problem (QPP))** *Given a quorum system $\mathcal{Q}$ over the universe U, along with an access strategy p, and also an undirected network $G = (V, E)$ with capacities $\mathsf{cap} : V \to \mathbb{R}^+$ and inter-point distances $d(\cdot, \cdot)$, find a placement $f : U \to V$ that (a) minimizes $\mathsf{Avg}_{v \in V}[\Delta_f(v)]$ (i.e., "low delay") subject to (b) $\mathsf{load}_f(v) \le \mathsf{cap}(v)$ for all $v \in V$ (i.e., "low load").*

[1] One can imagine choosing the input quorum system $\mathcal{Q}$ and access strategy p from the existing literature to achieve good load-balancing, say, or high availability, or any other desired criterion.

Our main result is the following:

THEOREM 1.2. *Let f^* be an optimal solution to the Quorum Placement Problem. Then, for any $\alpha > 1$, we can find in polynomial time a placement f with $\mathsf{load}_f(v) \le (\alpha + 1)\, \mathsf{cap}(v)$ for all $v \in V$, and for which*

$$\mathsf{Avg}_{v \in V}[\Delta_f(v)] \quad \le \quad \tfrac{5\alpha}{\alpha - 1}\, \mathsf{Avg}_{v \in V}[\Delta_{f^*}(v)] \quad (3)$$

Hence, e.g., we can find a map f that exceeds the capacities on the nodes by a factor of 4, but ensures that the delay is within a factor of 7.5 of the optimal delay of f^*.

We then go on to consider some well-known quorum systems, and show the following:

THEOREM 1.3. *Consider the Grid [5, 12] or Majority [8, 22] quorum systems on U with the access strategy p being the uniform distribution over all quorums. Given any graph G, we can find placements f such that $\mathsf{load}_f(v) \le \mathsf{cap}(v)$ for each $v \in V$, and where the average max-delay is at most 5 times the optimum among all such solutions.*

Finally, we continue with the study of quorum placement for another natural notion of access cost from a client to a quorum, the *total delay*, which captures the delay incurred if quorum elements are contacted sequentially. Specifically, if the vertex v accesses the quorum Q, then let $\gamma_f(v, Q) = \sum_{u \in Q} d(v, f(u))$ be the *total delay* for this access. Given access strategy p, the *expected total delay* for v is $\Gamma_f(v) = \sum_{Q \in \mathcal{Q}} p(Q)\, \gamma_f(v, Q)$. As before, we will be looking for a placement f to minimize $\mathsf{Avg}_{v \in V}[\Gamma_f(v)]$. Our main result for this measure is:

THEOREM 1.4. *We can find, in polynomial time, a placement $f : U \to V$ where $\mathsf{load}_f(v) \le 2\, \mathsf{cap}(v)$ at each node $v \in V$ and that has average delay $\mathsf{Avg}_{v \in V}[\Gamma_f(v)]$ at most the optimal delay among all placements satisfying $\mathsf{load}_f(v) \le \mathsf{cap}(v)$.*

1.3 Roadmap and Techniques

Let us give a brief roadmap of the paper, mentioning the ideas and techniques that we use along the way. In Section 3, we prove a crucial structural result that guides the rest of the paper. We show that, for any placement f that is a solution to the Quorum Placement Problem for max-delay, there exists a special node v_0, such that even if all the requests are routed through v_0, the average max-delay is at most 5 times that of f. (Hence the additional delay incurred by taking such a detour is not too large, a somewhat surprising fact.) Of course this is only a structural result: no practical algorithm would want to route all the requests through a single node, for fear of creating a bottleneck and a single point of failure.

However, we can now use this result to derive the following important consequence. The average delay of this "relay-via-v_0" routing strategy can be decomposed into two parts: **(i)** the average delay from the clients $v \in V$ to v_0 (which is a constant), and **(ii)** the delay from v_0 to a random quorum of $\mathcal{Q}$ chosen from p, which is just $\Delta_f(v_0)$. Hence, minimizing the overall average delay in this relaying strategy is equivalent to minimizing the average delay for the special case of the Quorum Placement Problem when v_0 is the only client in the system. This allows us to focus, for the rest of Sections 3 and 4 on this "single-source" version of QPP; we show that any solution to the Single-Source Quorum Placement Problem can be translated back to a solution

for Quorum Placement Problem with only a factor 5 loss in the average delay.

In Section 3, we formalize the Single-Source Quorum Placement Problem, and show it is NP-hard. We then present an approximation algorithm for it that achieves an average delay of at most $(\frac{\alpha}{\alpha-1})$ times the optimal, if we allow the load on each node to violate the given capacities by a factor of $(\alpha + 1)$. Combining this with our structural lemma (and hence incurring a loss of a factor of 5), we get the algorithm claimed in Theorem 1.2.

In Section 4, we present optimal solutions for the Single-Source Quorum Placement Problem for two well-known quorum systems, the **Grid** [5, 12] and the **Majority** [8, 22]. Combining this with the above reduction, we immediately get Theorem 1.3.

In Section 5, we address the total delay measure and prove Theorem 1.4. Finally, in section 6, we summarize and discuss extensions of our results.

2. RELATED WORK

Despite being over twenty years old, research on quorum systems remains an active and rich area; see, e.g., [1, 2, 3, 16, 17, 26] and the references therein. Previous work on quorum placement problems in graphs to minimize delays is scarcer; in particular, most previous work does not take into consideration network-oblivious measures such as load, and the natural trade-offs arising between delay and load. Specifically, Fu [7] introduced the following problem: given a graph $G = (V, E)$, find a quorum system $\mathcal{Q}$ over universe V to minimize $\mathsf{Avg}_{v\in V}[\min_{Q\in\mathcal{Q}} \delta(v, Q)]$, i.e., the average cost for each client to reach its "closest" quorum. That work presented optimal algorithms when G has certain characteristics, e.g., G is a tree, cycle or cluster network.

Problems of quorum design and placement on general graphs were then considered by Tsuchyia et al. [23], who gave an efficient algorithm to find $\mathcal{Q}$ so as to minimize $\max_{v\in V} \min_{Q\in\mathcal{Q}} \delta(v, Q)$, i.e., the maximum cost any client pays to reach its closest quorum. Kobayashi et al. [11] looked at the problem of designing quorums $\mathcal{Q}$ to minimize $\mathsf{Avg}_{v\in V}$ $[\min_{Q\in\mathcal{Q}} \delta(v, Q)]$. They gave a branch-and-bound algorithm for it, which could be evaluated only on topologies with up to 20 nodes due to its exponential running time, and they also conjectured that the problem is NP-hard. Following up on this work, Lin [14] showed that the problem is indeed NP-hard; this work, which directly motivated our research, also gives a 2-approximation for the problem.

At this point, let us mention that none of these works considered the load of the quorum system; indeed, Lin's 2-approximation [14] yields a quorum system with very high load—the output consists of only a *single* quorum containing a *single element* which is placed at a single node $v_0 \in V$ which minimizes $\sum_{v'\in V} d(v, v')$. Such a solution is not very desirable, since it eliminates the advantages (such as load dispersion and fault tolerance) of any distributed quorum-based algorithm. As discussed in the introduction, maintaining a low load and preserving this load dispersion capability is an essential requirement in the problems we study.

Independently of our work, Gilbert and Malewicz [9] consider a problem they call the "partial quorum deployment problem". As in all the problems we study, their problem also takes as inputs a graph $G = (V, E)$ and a quorum system $\mathcal{Q}$ over a universe U. However, they restrict the inputs so that $|\mathcal{Q}| = |V| = |U|$, and so that each client $v \in V$ selects only a single, distinct quorum to access. In this setting, they provide a polynomial-time algorithm to compute bijections $f : U \to V$ and $q : V \to \mathcal{Q}$ that minimize $\mathsf{Avg}_{v\in V}\gamma(v, f(q(v)))$, where $f(Q) = \{f(u)\}_{u\in Q}$. They also offer a number of negative results for other variations of the Quorum Placement problem, all of which are related to $\mathsf{Avg}_{v\in V}[\Gamma_f(v)]$. Our results for the same objective function (given in Section 5) generalize the scenario they consider: we weaken the restrictions on the inputs, and consider more general restrictions on the load of the system.

In more distantly related work, Carmi et al. [4] study the following problem, which we call the *geographic partition* problem: given a set $\mathcal{X}$ of n points in a closed region R of the plane, find a partition $\mathcal{Q}$ of $\mathcal{X}$ into clusters of size k so as to minimize $\max_{v\in R} \min_{Q\in\mathcal{Q}} \delta(v, Q)$. (Here, distances are in the plane.) They also address the issue of load balancing when the geographic partition $\mathcal{Q}$ is given: assuming that the clients are uniformly distributed across the region R, the problem is to find a partition of R into subregions of equal area such that each $Q \in \mathcal{Q}$ is contained in exactly one such subregion. Carmi et al. present efficient approximation algorithms for these problems, and using techniques in Dolev et al. [6] these can be utilized to implement intersecting quorums. However, this conversion does not preserve the delay properties of the underlying partition, and so does not solve the problem that we consider here (even in the plane).

3. MAXIMUM DELAY ACCESS COST

In this section we address the Quorum Placement Problem, and give our results for the *max-delay* access cost. Our first result is the following simple yet crucial structural result. Imagine that we are given a quorum placement $f : U \to V$ for the quorum system $\mathcal{Q}$. Then we find a special node v_0, such that routing all the requests to elements in $f(U)$ via the node v_0 causes the average access delay to be ≤ 5 times the delay when we route the requests to $f(U)$ along shortest paths. In other words, even though each message in the system takes a detour via v_0, the average delay does not increase by much, which we feel is a somewhat surprising fact.

Although this result seems to have no practical relevance (because we clearly don't want to route all the traffic through a single node), it gives us a way of accounting for, and approximately minimizing the average delay in the Quorum Placement Problem: we can split the delay $\Delta_f(v)$ for any vertex v when using this "relay-via-v_0" strategy into two components—the first corresponding to the delay in getting from v to v_0, and the other corresponding to the delay from v_0 to the quorums of $\mathcal{Q}$. But the former contribution, when averaging over all the clients, is a constant. Hence to minimize delay, it suffices to search for placements of $\mathcal{Q}$ that minimize the average delay incurred when the single node v_0 issues all the requests, which is $\Delta_f(v_0)$. To this end, we define the Single-Source Quorum Placement Problem at the end of Section 3.1.

We show that minimizing the delay in this new single-client problem is NP-hard for general quorum systems; the proof of this theorem appears in Section 3.2. We then give an algorithm in Section 3.3 which gives a placement that approximates the delay within a factor of $\frac{\alpha}{\alpha-1}$, but violates the load on each node by a factor of $\alpha + 1$. Combining this with the factor-5 loss in the reduction to the single-client case gives us Theorem 1.2. Finally, Section 4 gives efficient

algorithms for placing some well-known quorum system constructions when the access strategies yielding optimal load on them are used.

3.1 Reduction to the Single Client Case

The following structural lemma shows that there is a *single* node v_0 in the graph G such that even if all the messages to the quorum elements were sent via the node v_0, the access delay would not increase by more than a factor of 5.

LEMMA 3.1. *Consider any placement $f : U \to V$ of a quorum system $\mathcal{Q}$ on the nodes of G, and an access strategy p. Then there exists a vertex $v_0 \in V$ such that*

$$\mathsf{Avg}_{v\in V}[\sum_{Q\in\mathcal{Q}} p(Q)\left(d(v,v_0)+\delta_f(v_0,Q)\right)] \;\le\; 5\mathsf{Avg}_{v\in V}[\Delta_f(v)]. \tag{4}$$

PROOF. Before we begin the proof, note that the expression on the left in (4) is the average max-delay incurred if each message from v to the elements of Q first goes to v_0 (giving the $d(v, v_0)$ term) and then onto Q (giving the $\delta_f(v_0, Q)$ term).

For the proof, consider two vertices v and v' in V, and let us choose quorums Q and Q' independently at random from the distribution p. Recall that $\delta_f(v, Q) = \max_{u\in Q} d(v, f(u))$ is the maximum distance from v to any nodes in $f(Q)$. By the quorum intersection property, $Q \cap Q' \neq \emptyset$, and the triangle inequality, we get that $d(v, v') \le \delta_f(v, Q) + \delta_f(v', Q')$. Since the quorums Q and Q' were chosen from the distribution p independently, taking expectations over the distribution we get

$$d(v, v') \le \Delta_f(v) + \Delta_f(v').$$

Let v_0 be the node that minimizes $\Delta_f(v')$; that is, $v_0 = \text{argmin}_{v'\in V} \Delta_f(v')$. We then have:

$$d(v, v_0) \le \Delta_f(v) + \Delta_f(v_0) \le 2 \cdot \Delta_f(v) \tag{5}$$

(Note that, given an f, the node v_0 can be determined in time polynomial in $|V|$ just by trying all possible nodes v'.) We can now use the triangle inequality to bound the max-delay of messages sent via v_0 from the node v to the quorum Q (i.e., the "relay-via-v_0" strategy) by

$$\begin{aligned} d(v, v_0) + \delta_f(v_0, Q) &\le d(v, v_0) + d(v_0, v) + \delta_f(v, Q) \\ &\le 4 \cdot \Delta_f(v) + \delta_f(v, Q), \end{aligned} \tag{6}$$

where we have used (5) in the second line. Now taking expectations over p, and taking an average over $v \in V$, we get

$$\begin{aligned} &\mathsf{Avg}_{v\in V}[\sum_{Q\in\mathcal{Q}} p(Q)\left(d(v,v_0)+\delta_f(v_0,Q)\right)] \le \\ &4\ \mathsf{Avg}_{v\in V}[\Delta_f(v)] + \mathsf{Avg}_{v\in V}[\sum_{Q\in\mathcal{Q}} p(Q)\delta_f(v,Q)], \end{aligned} \tag{7}$$

which simplifies to the claimed expression (4) using the definition (2) of $\Delta_f(v)$. □

Hence, even if the messages sent from each node v to quorum elements are relayed via node v_0, the resulting average delay is still less than 5 times the optimal. Moreover, the expression on the left hand side of (4), which is the average delay of this "relay-via-v_0" strategy simplifies to

$$\mathsf{Avg}_{v\in V}[d(v, v_0)] + \Delta_f(v_0). \tag{8}$$

Hence, it makes sense to try and find a placement that minimizes $\Delta_f(v_0)$, and solve the following problem:

PROBLEM 3.2. **(Single-Source Quorum Placement Problem (SSQPP))** *Given a quorum system $\mathcal{Q}$ over a universe U, a graph $G = (V, E)$ with a special node v_0, an access strategy p_0 with which v_0 accesses quorums in $\mathcal{Q}$ and for each node $v \in V$ an upper bound $\mathsf{cap}(v)$ on the load it can support, find a placement $f : U \to V$ that (a) minimizes the average delay $\Delta_f(v_0)$ subject to (b) $\mathsf{load}_f(v) \le \mathsf{cap}(v)$ at each node.*

The following theorem summarizes the above discussion, and formalizes the reduction from the QPP to the Single-Source Quorum Placement Problem:

THEOREM 3.3. *There exists a node v_0 such that a placement $f : U \to V$ that is a β-approximation for the Single-Source Quorum Placement Problem (with source v_0) is also a $5\,\beta$-approximation to the general Quorum Placement Problem.*

PROOF. Consider the best placement f^* for the original Quorum Placement Problem instance, and find the node v_0 promised by Lemma 3.1 when given the placement f^*. Note that the placement f^* is also a solution for the Single-Source Quorum Placement Problem with special node v_0, and hence any β-approximate solution f to the Single-Source Quorum Placement Problem instance would have $\Delta_f(v_0) \le \beta\ \Delta_{f^*}(v_0)$. Thus the delay of a "route-via-v_0" strategy with this map f would have average delay

$$\mathsf{Avg}_{v\in V}[d(v, v_0)] + \beta\ \Delta_{f^*}(v_0) \le 5\beta\mathsf{Avg}_{v\in V}[\Delta_{f^*}(v)],$$

which follows from (4) and proves the result. □

Since we do not know the identity of the node v_0, we can run the Single-Source Quorum Placement Problem algorithm with each node in V, and pick the best placement among these.

3.2 NP-hardness of Problem 3.2

In this section, we show that the Single-Source Quorum Placement Problem is NP-hard, via a reduction from a classical NP-hard scheduling problem $1|prec|\sum w_j C_j$ [21].

DEFINITION 3.4. *The input to the problem $1|prec|\sum w_j C_j$ consists of n jobs $\{J_1, J_2, \ldots, J_n\}$ with job J_j having processing time T_j and weight w_j. Furthermore, there are precedence constraints given by a partial order $\prec$ on the jobs, such that if $J_i \prec J_j$, then any feasible schedule must put job J_i before J_j. The objective is to find a feasible schedule of the jobs on a single machine so that, if the completion time of job J_j is C_j, the weighted completion time $\sum w_j C_j$ is minimized.*

In fact, a theorem of Woeginger [25] implies that it suffices to consider only a special case of this scheduling problem:

THEOREM 3.5. [25] *The following statements are equivalent:*

(a) There exists a ρ-approximation for the general problem $1|prec|\sum w_j C_j$.

(b) There exists a ρ-approximation for the special case of $1|prec|\sum w_j C_j$ where every job has either $T_j = 0$ and $w_j = 1$, or $T_j = 1$ and $w_j = 0$, and where the existence of a precedence constraint $J_i \prec J_j$ implies that $T_i = 1$ and $w_i = 0$, and that $T_j = 0$ and $w_j = 1$.

The following theorem shows a polynomial time reduction of the scheduling problem $1|prec|\sum w_jC_j$ given in Theorem 3.5(b) to the Single-Source Quorum Placement Problem.

THEOREM 3.6. *The Single-Source Quorum Placement Problem 3.2 is NP-hard.*

PROOF. Consider an instance of the scheduling problem with jobs $\{J_1, J_2, \ldots, J_n\}$; let us reorder the jobs so that all the ones with zero weight appear before those with non-zero weight. Let there be m jobs with unit weight, and hence $\{J_1, J_2, \ldots, J_{n-m}\}$ have zero weight.

For each J_j with $T_j = 1$ (and hence $w_j = 0$), let us construct an element e_j in the universe U; we add an extra element e_0 to U—hence $|U| = n-m+1$. Let $0 < \epsilon < 1$ be a constant to be fixed later. The quorums and access strategy p are defined thus:

- For each of the m unit-weight jobs J_j (with $T_j = 0$ and $w_j = 1$), define a quorum $Q_j = \{e_0\} \cup \{e_{j'} \in U \mid J_{j'} \prec J_j\}$. Each of these quorums Q_j is accessed with probability $p(Q_j) = \frac{\epsilon}{m}$. (Call these the *type-1 quorums*.)
- For each element $u \in U$ such that $u \neq e_0$, define a quorum $Q_u = \{u, e_0\}$. Each of these quorums Q_u is accessed with probability $p(Q_u) = \frac{1-\epsilon}{n-m}$. (Call these *type-2 quorums*.)

Note that all the quorums intersect in e_0; furthermore, since there are m quorums Q_j and $(n-m)$ quorums Q_u, the access strategy p is indeed a probability distribution over the quorum system $\mathcal{Q}$. Note that the load on element e_0 is $\mathsf{load}(e_0) = 1$; for any other element $u \in U$, if u belongs to $n_u \leq m$ type-1 quorums Q_j, then its load is $\mathsf{load}(u) = \epsilon \cdot \frac{n_u}{m} + (1-\epsilon) \cdot \frac{1}{n-m}$. Choosing ϵ such that $\epsilon < \frac{(1-\epsilon)}{n-m}$, the load for any $u \in U \setminus \{e_0\}$ is $\mathsf{load}(u) \in [\frac{1-\epsilon}{n-m}, \frac{2(1-\epsilon)}{n-m})$.

Finally, the graph $G = (V, E)$ is just a path with a node $v_j \in V$ for each element $e_j \in U$ (hence ensuring that $|V| = |U| = n-m+1$), and edges (v_i, v_{i+1}) for $0 \leq i < n-m$ of unit length. The capacity of each node v_j with $j \neq 0$ is $\mathsf{cap}(v_j) = \frac{2(1-\epsilon)}{n-m} - \epsilon$, and the capacity of v_0 is $\mathsf{cap}(v_0) = 1 = \mathsf{load}(e_0)$. This completes the construction of an instance of the Single-Source Quorum Placement Problem.

Let us note some useful properties of the construction: since the capacity of any node $v_j \neq v_0$ is strictly less than $1 = \mathsf{load}(e_0)$, the element e_0 can only be placed on v_0. Moreover, since the load of any element $u \neq e_0$ lies in $[\frac{1-\epsilon}{n-m}, \frac{2(1-\epsilon)}{n-m})$, and the capacity of any $v_j \neq v_0$ is strictly less than $2 \times \frac{1-\epsilon}{n-m}$, any feasible placement must place exactly one element in U on a node of G. Furthermore, any permutation of the elements in $U - \{e_0\}$ can be placed on the vertices $V - \{v_0\}$. Thus, any placement f can be converted into a schedule in the natural way: if $f(e_j) = v_t$ for $e_j \neq e_0$ we schedule the zero-weight job J_j at time t. We also schedule unit-weight jobs $J_{j'}$ at the earliest time possible subject to the precedence constraints.

It suffices now to show that the optimal solution to the SSQPP instance gives us an optimal solution to the scheduling problem. Denote by π_f the schedule constructed from a placement f as described above. Let t_j be the time when zero-weight job J_j is scheduled and let $t_{j'}$ be the time when unit-weight job $J_{j'}$ is scheduled. The completion time of schedule π_f is then: $\mathsf{cost}(\pi_f) = \sum w_jC_j = \sum_{j':w_{j'}=1} t_{j'}$. The average delay of the placement f is:

$$\begin{aligned}\Delta_f(v_0) &= \sum_{\text{type-1 } Q_j} p_0(Q_j)t_j + \sum_{\text{type-2 } Q_{j'}} p_0(Q_{j'})t_{j'} \\ &= \sum_{\text{type-1 } Q_j} \frac{\epsilon}{m} t_j + \sum_{\text{type-2 } Q_{j'}} \frac{1-\epsilon}{n-m} t_{j'} \\ &= \frac{\epsilon}{m} \cdot \mathsf{cost}(\pi_f) + \frac{1-\epsilon}{n-m} \cdot \sum_{i=1}^{n-m} i\end{aligned}$$

Now it becomes easy to see that the completion time of the schedule π_f is minimized if and only if the average delay of the placement f is also minimized. □

3.3 A Linear Program Rounding Solution

In light of the intractability result in Theorem 3.6, our goal is now to find an approximation algorithm for the Single-Source Quorum Placement Problem. To this end, we formulate Problem 3.2 as an integer linear program, consider its linear programming (LP) relaxation, and round this LP to get an integral solution. Unfortunately, this linear program has a large integrality gap of $O(\sqrt{n})$—see Appendix A for the definition of this concept, as well as the example showing the integrality gap. However, we show that one can still get a 2-approximation algorithm from it by a *resource augmentation* argument [10], i.e., if we allow ourselves to violate the capacity at any node v by a small factor.

Some useful notation: since we are interested only in distances from the node v_0, let us rename nodes as $\{v_0, v_1, v_2, \ldots, v_{n-1}\}$ so that $d(v_0, v_1) \leq d(v_0, v_2) \leq \cdots \leq d(v_0, v_{n-1})$. Let us also denote $d(v_0, v_t)$ by d_t, and hence $0 = d_0 \leq d_1 \leq \ldots \leq d_{n-1}$. Let $f^* : U \to V$ be an optimal solution to Problem 3.2; i.e., a placement that minimizes $\Delta_{f^*}(v_0)$ subject to the constraints $\sum_{u:f^*(u)=v} \mathsf{load}(u) \leq \mathsf{cap}(v)$.

To write an integer linear programming formulation for the problem, let us denote by x_{tu} the indicator for whether the element $u \in U$ is placed on the physical node v_t. Similarly, given $v_t \in V$ and $Q \in \mathcal{Q}$, the variable $x_{tQ} = 1$ indicates that all elements $u \in Q$ are placed on some subset of the nodes $\{v_0, \ldots, v_t\}$. The LP is given by:

$$\text{minimize } Z^* = \textstyle\sum_Q p_0(Q) \sum_{t=1}^n d_t \, x_{tQ} \tag{9}$$

$$\textstyle\sum_t x_{tu} = 1 \qquad \forall\, u \in U \tag{10}$$

$$\textstyle\sum_t x_{tQ} = 1 \qquad \forall\, Q \in \mathcal{Q} \tag{11}$$

$$\textstyle\sum_u \mathsf{load}(u) x_{tu} \leq \mathsf{cap}(v_t) \qquad \forall\, v_t \in V \tag{12}$$

$$x_{tu} = 0 \qquad \forall\, u \in U,\ \forall\, v_t \in V \text{ s.t. } \mathsf{load}(u) > \mathsf{cap}(v_t) \tag{13}$$

$$\textstyle\sum_{s \leq t} x_{sQ} \leq \sum_{s \leq t} x_{su} \quad \forall\, u \in Q,\ \forall\, v_t \in V,\ \forall\, Q \in \mathcal{Q} \tag{14}$$

The constraint (10) implies that each element u is assigned to one node, and (12) implies that no node v_t has too much load assigned to it. Constraints (14) and (11) ensure that if $x_{tQ} = 1$ then all the elements in Q are indeed placed on some subset of $\{v_0, \ldots, v_t\}$, and that there is indeed one such value of t. Finally, (13) ensures that no node v_t is assigned an element u with $\mathsf{load}(u)$ more than v_t's capacity $\mathsf{cap}(v_t)$.

Note that if the variables are all either 0 or 1, we could get the placement $f : U \to V$ by setting $f(u) = v_t \iff x_{tu} = 1$; indeed, it is easy to see that this would be an exact

formulation of the Single-Source Quorum Placement Problem. However, finding such an integral solution is NP-hard, and thus we consider the LP relaxation where the variables can take fractional values between 0 and 1: such a solution can be obtained in polynomial time. Since we have taken a relaxation of the problem, it follows that $Z^* \leq \Delta_{f^*}(v_0)$.

We now show how to round the fractional solution x to obtain a map f which has an average delay at most $2Z^* \leq 2\Delta_{f^*}(v_0)$, but where the load at any node v_t is $\mathsf{load}(v_t) = \sum_{u:f(u)=v_t} \mathsf{load}(u) \leq 3\,\mathsf{cap}(v_t)$. One can then generalize this result and trade off the losses in the delay and load seamlessly to get the following:

THEOREM 3.7. *For any $\alpha > 1$, we can find a solution $f : U \to V$ to the Single-Source Quorum Placement Problem, with delay $\Delta_f(v_0) \leq \frac{\alpha}{\alpha-1}$ and load on any node $v \in V$ satisfying $\sum_{u:f(u)=v} \mathsf{load}(u) \leq (\alpha+1)\,\mathsf{cap}(v)$.*

Note that the proof below is just the case $\alpha = 2$ of this theorem; the extension is not difficult, and we postpone the details until after Theorem 3.12.

3.3.1 Rounding the Fractional LP Solution

The process of rounding the fractional solution to obtain the integral solution (and hence the map f) consists of two conceptual steps. Let us give the high-level sketch before we give the details.

Filtering. In this step, we alter the LP solution to obtain a "good" (fractional) solution in which no element u is fractionally assigned to nodes that are "too far away" from v_0. Formally, after this step, if S_u is the set of nodes v_t such that $x_{tu} > 0$, then any map f satisfying $f(u) \in S_u$ will still allow us to guarantee that $\Delta_f(v_0) \leq 2Z^* \leq 2\Delta_{f^*}(v_0)$.

Rounding. We now view the good fractional solution obtained from the above step as a solution to the generalized assignment problem (GAP) and use a rounding procedure for this problem to convert the fractional solution into an integral solution such that the total load assigned to any node v_t is at most $3\,\mathsf{cap}(v_t)$.

Filtering. For each element u, let $\widehat{x}_{tu}$ be the largest possible value subject to the constraints that $\widehat{x}_{tu} \leq 2x_{tu}$ and $\sum_{t \leq s} \widehat{x}_{tu} \leq 1$. More precisely we set $\widehat{x}_{tu} = 2x_{tu}$ for all t's such that $\sum_{s \leq t} x_{su} < \frac{1}{2}$ and $\widehat{x}_{tu} = 1 - \sum_{s<t} \widehat{x}_{su}$ for the first t such that $\sum_{s \leq t} x_{su} > \frac{1}{2}$. We change the values $\widehat{x}_{tQ}$ similarly. (Intuitively, we are "moving mass" to the lower values of t.) Note that these new values $\widehat{x}$ satisfy (10); and they violate (12) by a factor of at most 2. Moreover, for any values of t and $u \in Q$, $\sum_{s \leq t} x_{sQ} \leq \sum_{s \leq t} x_{su}$: hence when going from x to $\widehat{x}$, either both the left and right sides of the inequality double, or the right side becomes equal to 1—in both cases, (14) holds. Note that the modified value of the objective function satisfies $\sum_Q p_0(Q) \sum_{t=1}^n d_t\, \widehat{x}_{tQ} \leq Z^*$.

We now formalize the statement that no element u is assigned to v_t which is "too far". Consider the objective function (9) of the LP: define $\overline{D}_Q = \sum_t d_t x_{tQ}$ for any quorum Q, and thus the LP value is $Z^* = \sum_Q p_0(Q)\overline{D}_Q$.

CLAIM 3.8. *All elements of quorum Q are fractionally assigned to nodes v_t with $d_t = d(v_0, v_t)$ at most $2\,\overline{D}_Q$. In other words, if $\widehat{x}_{tQ} > 0$ for some t, then $d_t \leq 2\,\overline{D}_Q$.*

PROOF. This is just Markov's inequality, but here is the longer explanation. Look at the largest value t for which $\widehat{x}_{tQ} > 0$; this must be a value such that $\sum_{s<t} x_{sQ} < \frac{1}{2}$ and $\sum_{s \leq t} x_{sQ} \geq \frac{1}{2}$. If $d_t > 2\overline{D}_Q$, then Q is assigned to values larger than $d_t > 2\overline{D}_Q$ for at least a fraction of $(1 - \sum_{s \leq t} x_{sQ}) > \frac{1}{2}$, which violates the fact that $\overline{D}_Q$ is the average $\sum_s x_{sQ} d_s$. □

LEMMA 3.9. *For any element $u \in U$, let $S_u = \{v_t \in V \mid \widehat{x}_{tu} > 0\}$. Then for any map f that places elements $u \in U$ on nodes in the corresponding set S_u, we have that $\Delta_f(v_0) \leq 2Z^*$.*

PROOF. If $f(u) \in S_u$ then $\widehat{x}_{tu} > 0$ and so Claim 3.8 says that $\delta_f(v_0, Q) = \max_{u \in Q} d(v_0, f(u)) \leq 2\,\overline{D}_Q$. Therefore $\Delta_f(v_0) = \sum_Q p_0(Q)\delta_f(v_0, Q) \leq \sum_Q p_0(Q) \times 2\,\overline{D}_Q \leq 2Z^*$. □

Rounding. We will now view the modified solution $\widehat{x}$ for the LP as a fractional solution to a suitable instance of the so-called *Generalized Assignment Problem* (GAP), and use techniques for that problem to round the fractional $\widehat{x}$'s to an integral solution that satisfies the assumptions of Lemma 3.9.

DEFINITION 3.10 (**GAP**). *The GAP problem takes as input a set U of "jobs" and a set V of "machines", and for each $(j, i) \in U \times V$ two positive values: c_{ij} being the cost of assigning job j to machine i, and p_{ij} being the load imposed by such an assignment to machine i. The output is an assignment f of the jobs to the machines, of minimum cost $\sum_{j \in U} c_{f(j)j}$, subject to constraints on the load $\sum_{j \in f^{-1}(i)} p_{ij} \leq T_i, \forall i \in V$, given constants $T_i \in \mathbb{R}^+$ for each $i \in V$.*

Consider the following natural LP relaxation of GAP:

$$\text{minimize } Y^* = \sum_{j \in U} \sum_{i \in V} c_{ij} y_{ij} \tag{15}$$

$$\sum_{j \in U} p_{ij} y_{ij} \leq T_i \qquad \forall\, i \in V \tag{16}$$

$$\sum_{i \in V} y_{ij} = 1 \qquad \forall\, j \in U \tag{17}$$

$$y_{ij} \geq 0 \qquad \forall\, j \in U, i \in V \tag{18}$$

This relaxation was studied by Lenstra et al. [13] and Shmoys and Tardos [20], who proved the following result. (Here $p_i^{\max}$ is the largest load of any job assigned to machine i.)

THEOREM 3.11. [20] *Any fractional solution for the LP relaxation of GAP can be rounded into an integral solution with cost no more than Y^*, with the load on machine i being at most $T_i + p_i^{\max} \leq 2\,T_i$.*

We can use this powerful result to round our LP with the new variables $\widehat{x}$ by the following translation: the elements $u \in U$ correspond to the jobs $j = u$; the nodes $v_t \in V$ correspond to the machines $i = t$; the load p_{tu} for machine t and job u is $\mathsf{load}(u)$ if $\widehat{x}_{tu} > 0$ and $p_{tu} = \infty$ otherwise; the cost c_{tu} is the delay d_t; finally, the upper bound T_t for machine t is $2\mathsf{cap}(v_t)$.

Since we ensure that no element u can be fractionally assigned to node v_t if $\mathsf{load}(u) > \mathsf{cap}(v_t)$, this implies that the solution produced by applying Theorem 3.11 to the $\widehat{x}$'s places load at most $T_t + p_t^{\max} \leq 2\mathsf{cap}(v_t) + \mathsf{cap}(v_t) = 3\mathsf{cap}(v_t)$. Hence the capacity is violated by at most a factor of 2, which implies the following theorem:

THEOREM 3.12. *We can find a solution $f : U \to V$ to the Single-Source Quorum Placement Problem, where the delay $\Delta_f(v_0)$ is at most twice the LP optimum $Z^* \leq \Delta_{f^*}(v_0)$. Furthermore, the load on any node $v_t \in V$ is violated by at most a factor of three; i.e., $\sum_{u:f(u)=v_t} \mathsf{load}(u) \leq 3\,\mathsf{cap}(v_t)$.*

To obtain Theorem 3.7, we only need to change the factor of 2 from the filtering step above to an arbitrary $\alpha > 1$. Variables $\widehat{x}_{tu}$ become the largest possible, subject to the constraints $\widehat{x}_{tu} \leq \alpha x_{tu}$ and $\sum_{t \leq s} \widehat{x}_{tu} \leq 1$. This will increase the load on each node v_t for which $\widehat{x}_{tu} > 0$ by no more than a factor of α. With the additional loss from GAP we obtain the bound of $(\alpha+1)\mathsf{cap}(v)$ on the load of each node $v \in V$. The delay of any node v_t on which some element of quorum Q is placed (i.e., for which $\widehat{x}_{tQ} > 0$) becomes $d_t \leq \frac{\alpha}{\alpha-1}\overline{D}_Q$. The factor of $\frac{\alpha}{\alpha-1}$ propagates further through Lemma 3.9 leading to the bound on delay claimed in Theorem 3.7.

4. OPTIMAL LAYOUTS FOR SPECIFIC CONSTRUCTIONS

In this section we address the Single-Source Quorum Placement Problem for some specific quorum systems, and give explicit placements that respect the capacities $\mathsf{cap}(v)$ at the nodes while minimizing the average delay $\Delta_f(v_0)$. The specific quorum systems considered here are the well-known **Grid** [5, 12] and the **Majority** [8, 22] quorum systems.

Note that the results here give us Theorem 1.3, since we can use the reduction of the Quorum Placement Problem to the Single-Source Quorum Placement Problem from Section 3.1 (with the attendant loss of a factor of 5 in the delay).

4.1 The Grid Construction

Consider the *Grid* quorum system [5, 12] on a universe U of k^2 elements. The k^2 elements are laid out on a k by k square grid M, and each quorum $Q \in \mathcal{Q}$ is formed by taking all the elements from some row and some column of M. Hence each quorum has $2k-1$ elements, and there are k^2 quorums in $\mathcal{Q}$. We assume that p_0 is the uniform access strategy, since this yields the optimal load for the Grid [18]. Due to this uniformity, we can rephrase the objective function $\Delta_f(v_0)$ as $\sum_{i=1}^{k^2} \max_{u \in Q_i}\{d(v_0, f(u))\}$.

For simplicity, let us consider the case where the capacity $\mathsf{cap}(v)$ of each node $v \in V$ is equal to the $\mathsf{load}(u)$ of any element $u \in U$ (which is the same for all elements $u \in U$ when the uniform access strategy is being used). We can easily extend our results to the general case by suppressing nodes with capacity less than $\mathsf{load}(u)$ and making multiple copies of nodes with a capacity large enough to fit multiple amounts of $\mathsf{load}(u)$ (this is equivalent to greedily packing amounts of $\mathsf{load}(u)$ into nodes with capacity $\mathsf{cap}(v) \geq \mathsf{load}(u)$). The problem then becomes one of matching U to the k^2 nearest nodes to v_0; let $\tau_1 \geq \ldots \geq \tau_{k^2}$ be the distances from v_0 to these k^2 nodes to v_0 in decreasing order (i.e., the distances $d_1, \ldots, d_{k^2}$ in reverse order).

A convenient way of visualizing a placement f is to look at a $k \times k$ matrix M where each entry is one of the $\tau_1, \ldots, \tau_{k^2}$ distances; the correspondence between such matrices and placements f is given by setting $f((i,j)) = v \iff M_{ij} = d(v_0, v)$, breaking ties arbitrarily. The problem is now to place the values $\tau_1, \ldots, \tau_{k^2}$ in a $k \times k$ matrix M that minimizes the sum over all quorums of the maximum distance τ_i in each quorum.

The general strategy is to place the largest l^2 distances on the top-left $l \times l$ square of M. The next l distances, (i.e.,$\tau_{l^2+1}, \tau_{l^2+2}, \ldots, \tau_{l^2+l}$) are placed on positions $M_{1,l+1}$, $M_{2,l+1}, \ldots, M_{l,l+1}$, and the $l+1$ after them (i.e., $\tau_{l^2+l+1}, \ldots,$ τ_{l^2+2l+1}) on cells $M_{l+1,1}, \ldots, M_{l+1,l+1}$. This gets us from a $l \times l$ square to a $(l+1) \times (l+1)$ square, and having started with τ_1 in $M_{1,1}$, we can complete the placement inductively. In Appendix B, we give a proof that this intuitive strategy is optimal.

4.2 The Majority Construction

We now consider the following simple generalization of the well-known Majority construction. Given a universe U of size n and a parameter $t \geq \lceil \frac{n}{2} \rceil$, the quorum system consists of all the subsets of U of size t. We claim that any placement of this quorum system on the nodes of the graph has the same average delay under the uniform access strategy, which is

$$\frac{1}{\binom{n}{t}} \times \sum_{i=1}^{n-t+1} \tau_i \times \binom{n-i}{t-1}. \tag{19}$$

This is easy to see, since there are $\binom{n-1}{t-1}$ quorums containing τ_1, $\binom{n-2}{t-1}$ quorums containing τ_2 but not τ_1, $\binom{n-3}{t-1}$ quorums containing τ_3 but not the preceding two, and so on until there is a single quorum containing τ_{n-t+1} but not $\tau_1, \ldots, \tau_{n-t}$.

5. TOTAL DELAY ACCESS COST

In this section, we present a polynomial time algorithm for the problem of minimizing the *total-delay* objective function $\mathsf{Avg}_{v \in V}[\Gamma_f(v)]$ introduced in Section 1, where the access cost from a client v to a quorum Q is the sum of distances from v to all the elements of the quorum Q. This turns out to be a somewhat more tractable problem, and we can directly use techniques based on the Generalized Assignment Problem (cf. Definition 3.10 and Theorem 3.11) to give us a placement f with total-delay within a factor of two of optimum.

THEOREM 5.1. *Consider a quorum system $\mathcal{Q}$ over a universe U with an access strategy p, and an undirected graph $G = (V, E)$ with its associated metric d and capacities $\mathsf{cap}(v)$ for each $v \in V$. If $f^* : U \to V$ is a placement that minimizes $\mathsf{Avg}_{v \in V}[\Gamma_f(v)]$ subject to $\mathsf{load}_{f^*}(v) \leq \mathsf{cap}(v)$, then we can find in polynomial time, a placement f with $\mathsf{load}_f(v) \leq 2\,\mathsf{cap}(v)$ on $v \in V$, and where*

$$\mathsf{Avg}_{v \in V}[\Gamma_f(v)] \quad \leq \quad \mathsf{Avg}_{v \in V}[\Gamma_{f^*}(v)] \tag{20}$$

I.e., the placement minimizes the average access cost, but violates the capacity of each node by at most a factor of two.

PROOF. Recall the LP relaxation 15-18 of GAP and Theorem 3.11. We will reduce our problem to the GAP problem, which will allow us to use the rounding technique of Theorem 3.11 to achieve our results.

In our problem, each element corresponds to a job in GAP (and hence we replace all j's by u's), and each graph node corresponds to a machine (thus replacing i's by v's). Also, if an element $u \in U$ is assigned to a graph node $v \in V$, its contribution to the load on v is $\mathsf{load}(u)$, and thus we set $p_{vu} = \mathsf{load}(u)$. Furthermore, the contribution to the average total-delay is $\frac{1}{n}\sum_{v' \in V}\sum_{Q: Q \ni u} p(Q) d(v', v)$, and this we set to be the "cost" c_{vu}. Of course, we set the upper bound T_v for each $v \in V$ to be the capacity $\mathsf{cap}(v)$.

We can now solve the GAP LP, and use the rounding result to ensure that the cost of the resulting solution is no more than $\mathsf{Avg}_{v \in V}[\Gamma_{f^*}(v)]$, and the load placed on any node is at most $2T_v = 2\mathsf{cap}(v)$, thus proving the theorem. □

6. SUMMARY AND DISCUSSION

In this paper we have introduced problems requiring the placement of quorums in a network so as to (approximately) minimize the average delay that clients incur to contact quorums, while (approximately) limiting the load each network node suffers to a predefined capacity. As quorums underlie numerous distributed algorithms, we believe our results are a step toward the use of such algorithms in wide-area networks, where different placements can result in varied delays for quorum access.

Numerous extensions of our results are possible. For example, a more general formulation of our Quorum Placement Problem allows clients to use different access strategies when contacting a quorum. We remark here that the proof of Lemma 3.1 still holds in this more general case. Furthermore, suppose that each client $v \in V$ has its own access strategy p_v. Assigning to each node an access strategy equal to the average of all the p_v's achieves the same average delay as the left-hand side of (4). Hence Theorem 1.2 holds for this more general version of the problem as well.

Another important observation regards the access rates made by different clients when accessing the quorum system. For the sake of simplicity we assumed these to be uniform. We emphasize here however, that our results hold even when clients use different access rates when contacting a quorum system.

7. REFERENCES

[1] Y. Amir and A. Wool. Optimal availability quorum systems: theory and practice. *Inform. Process. Lett.*, 65(5):223–228, 1998.

[2] R. A. Bazzi. Planar quorums. *Theoret. Comput. Sci.*, 243(1-2):243–268, 2000.

[3] R. A. Bazzi. Access cost for asynchronous byzantine quorum systems. *Dist. Comp.*, 14(1):41–48, 2001.

[4] P. Carmi, S. Dolev, S. Har-Peled, M. J. Katz, and M. Segal. Geographic quorum system approximations. *Algorithmica*, 41(4):233–244, 2005.

[5] S. Y. Cheung, M. H. Ammar, and M. Ahamad. The grid protocol: A high performance scheme for maintaining replicated data. *Knowledge and Data Engineering*, 4(6):582–592, 1992.

[6] S. Dolev, S. Gilbert, N.A. Lynch, A.A. Shvartsman, and J.L. Welch. Geoquorums: Implementing atomic memory in mobile *d hoc* networks. In *DISC*, pages 306–320, 2003.

[7] A.W. Fu. Delay-optimal quorum consensus for distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 8(1):59–69, 1997.

[8] D. K. Gifford. Weighted voting for replicated data. In *Proceedings of the 7th ACM Symposium on Operating Systems Principles (SOSP)*, pages 150–162, 1979.

[9] S. Gilbert and G. Malewicz. The quorum deployment problem. In *8th International Conference on Principles of Distributed Systems (OPODIS'04)*, 2004.

[10] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.

[11] N. Kobayashi, T. Tsuchiya, and T. Kikuno. Minimizing the mean delay of quorum-based mutual exclusion schemes. *Journal of Systems and Software*, 58(1):1–9, 2001.

[12] A. Kumar, M. Rabinovich, and R. K. Sinha. A performance study of general grid structures for replicated data. In *Proceedings 13th International Conference on Distributed Computing Systems*, pages 178–185, 1993.

[13] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming*, 46(3, (Ser. A)):259–271, 1990.

[14] X. Lin. Delay optimizations in quorum consensus. In *Proceedings of the 12th International Symposium on Algorithms and Computation*, pages 575–586. Springer-Verlag, 2001.

[15] M. Maekawa. A $\sqrt{n}$ algorithm for mutual exclusion in decentralized systems. *ACM Transactions on Computer Systems*, 3:145–159, 1985.

[16] D. Malkhi, M. K. Reiter, and A. Wool. The load and availability of Byzantine quorum systems. *SIAM J. Comput.*, 29(6):1889–1906 (electronic), 2000.

[17] D. Malkhi, M. K. Reiter, A. Wool, and R. N. Wright. Probabilistic quorum systems. *Inform. and Comput.*, 170(2):184–206, 2001.

[18] M. Naor and A. Wool. The load, capacity, and availability of quorum systems. *SIAM J. Comput.*, 27(2):423–447 (electronic), 1998.

[19] D. Peleg and A. Wool. Crumbling walls: A class of practical and efficient quorum systems. *Distributed Computing*, 10(2):87–97, 1997.

[20] D. B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Math. Programming*, 62(3, Ser. A):461–474, 1993.

[21] J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22-35, 1978.

[22] R. H. Thomas. A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems*, 4(2):180–209, 1979.

[23] T. Tsuchiya, M. Yamaguchi, and T. Kikuno. Minimizing the maximum delay for reaching consensus in quorum-based mutual exclusion schemes. *IEEE Transactions on Parallel and Distributed Systems*, 10(4):337–345, 1999.

[24] V. V. Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, 2001.

[25] G. J. Woeginger. On the approximability of average completion time scheduling under precedence constraints. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP' 2001)*, LNCS 2076, pages 862–874. Springer, 2001.

[26] H. Yu. Signed quorum systems. In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing*, pages 246–255. ACM Press, 2004.

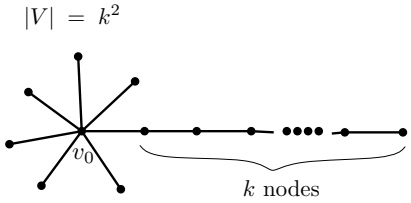

Figure 1: Graph with k^2 nodes

APPENDIX

A. INTEGRALITY GAP OF THE LP

In this section, we show that the integrality gap of the LP relaxation (9–14) is very large—even for distances that arise from a simple unweighted graph. This shows that that we cannot use this LP relaxation to bound the delay if we do not relax the node capacities $\mathsf{cap}(v)$.

Let us recall the definition of *Integrality Gap* (see, e.g., [24]). Given a linear programming relaxation for a minimization problem Π, let $LP(I)$ be the objective function value of an optimal fractional solution for the LP-relaxation. Let $OPT(I)$ denote the cost of an optimal (integral) solution for instance I. We define the *integrality gap* (or *integrality ratio*) of the LP-relaxation to be $\sup_I \frac{OPT(I)}{LP(I)}$.

CLAIM A.1. *The LP (9)- (14) has an integrality gap of at least n for general metric spaces, and at least $\sqrt{n}$ even for metric spaces induced by unweighted graphs.*

PROOF. For the former result, consider the following instance: the quorum system $\mathcal{Q}$ has only one quorum Q containing all the n elements. The values d_t are 1 for all $0 \le t < n-1$, and $d_{n-1} = M \gg 1$. In this case the only integral solution has $\Delta_f(v_0) = M$. However, we can set $x_{tQ} = x_{tu} = 1/n$ for all $u \in U$ and $0 \le t < n$; the LP objective function (9) is now just $\frac{1}{n}\sum_t d_t = \frac{n-1+M}{n} \approx \frac{M}{n}$ for $M \gg n$, and hence the integrality gap is about n.

Note that some of the edges of G have non-unit length in the above example; let us now present our result for the case when the underlying graph has all edges of unit length. Again assume that we have a single quorum Q containing all the elements, and consider the graph from Figure 1 with $n = k^2$ nodes. Setting the values d_i to be the distances from v_0 to all the other nodes of the graph G, we have $d_i = 1$ for $2 \le i \le n-k+1$, $d_{n-k+2} = 2$, $d_{n-k+3} = 3$, $\ldots$, $d_n = k$. In this case, the only integral solution has average delay k. However, if we set $x_{tQ} = x_{tj} = 1/n$ for all $1 \le t, j \le n$, the LP objective function has a value of

$$(n-k+1)\,\frac{1}{n} + 2\,\frac{1}{n} + \cdots + k\,\frac{1}{n} = \frac{1}{n}\left(n-k+\frac{k(k+1)}{2}\right) \approx \frac{3}{2}.$$

Thus, the integrality gap is at least $O(k) = O(\sqrt{n})$. □

B. DETAILS OF THE GRID LAYOUT

THEOREM B.1. *The placement f given in Section 4.1 for the $k \times k$ grid is an optimal solution for the Single-Source Quorum Placement Problem when the uniform access strategy is used.*

PROOF. We start with any optimal placement g and perform a number of transformations to it that do not increase its cost, at the end of which g will look as if obtained by using our strategy. This will prove that our placement strategy is optimal.

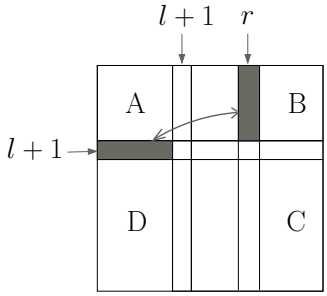

Figure 2: Partition of matrix M by row $l+1$ and column $l+1$ into regions A, B, C, D

Recall that the cost of a placement for the Single-Source Quorum Placement Problem is the average delay from the client v_0 to all the quorums in the quorum system. For the case of the Grid quorum system, the delay from v_0 to a quorum Q_{ij} formed by taking the union of the elements in row i and column j, is the maximum distance $\tau_{max_{ij}}$ placed by the map g on one of the cells of row i or column j of M. Since $M_{i,j}$ uniquely determines quorum Q_{ij}, we also assign to cell $M_{i,j}$ a cost equal to the delay of quorum Q_{ij}: $\mathsf{cellcost}(M_{i,j}) = \delta_g(v_0, Q_{ij}) = \tau_{max_{ij}}$. Thus the average delay of the placement g can also be written as $\Delta_g(v_0) = \frac{1}{k^2}\sum_{i,j=1}^{k} \mathsf{cellcost}(M_{i,j})$.

Let us begin with several simple observations. First, any swapping of rows or columns of M does not change the cost of the placement g. Second, a swap of two elements $\tau_i \le \tau_j$ placed at the intersection of two rows i and j with the same column, does not increase the cost of the placement g if τ_j is less than the maximum distance placed on row i. A similar statement can be made for elements placed at the intersection of two columns with one row.

Consider now the placement g. By swapping rows or columns we can bring element τ_1 on position $(1,1)$ of M without changing the cost of the placement. Now assume that we have changed the position of elements $\tau_1, \ldots, \tau_{l^2}$ starting from the initial placement g to arrive at a placement consistent with our strategy and such that the cost has not increased. This implies that $\tau_1, \ldots, \tau_{l^2}$ are placed in the top left $l \times l$ square. We now show how to reposition $\tau_{l^2+1}, \ldots, \tau_{(l+1)^2}$ according to our strategy without increasing the cost.

To make the exposition easier to follow we use the following notation: row $l+1$ and column $l+1$ partition matrix M into four regions A, B, C, D, where A and B are the intersections of the first l rows with the first l columns and the last $k-l$ columns respectively, while D and C are intersections of the last $k-l$ rows with the first l columns and the last $k-l$ columns respectively, as in Figure 2. At the end of the previous step, each of the elements $\tau_{l^2+1}, \ldots, \tau_{(l+1)^2}$ can only be in one of the B, C, D regions. Note that all of them are less than the maximums of rows 1 through l and columns 1 through l, since we placed elements $\tau_1, \ldots, \tau_{l^2}$ in region A.

Now look at element τ_{l^2+1}. If it is in one of the B or D regions, we can easily bring it on a cell adjacent to A by swapping rows or columns. If it is in region C, we can

swap it with any of the elements from B that are on the same column without increasing the cost, since any of the first l rows has a maximum greater than τ_{l^2+1} and τ_{l^2+1} is greater than any of the elements from its column. After this move we can swap columns as before to bring τ_{l^2+1} to a cell adjacent to A. Let us assume, without loss of generality, that the position of this cell is $M_{1,l+1}$.

Consider now element τ_{l^2+2}. Assume position $M_{2,l+1}$ is occupied by $\tau_i \leq \tau_{l^2+2}$. Then swapping the two elements will not increase the cost, since placing τ_{l^2+2} on $M_{2,l+1}$ does not change the cost, while placing $\tau_i \leq \tau_{l^2+2}$ on the old position of τ_{l^2+2} might only decrease the cost. By a similar argument one can bring $\tau_{l^2+3}, \ldots, \tau_{l^2+l}$ to the right positions.

Finally, look at element τ_{l^2+l+1}. If it is in regions C or D we can bring it on a cell adjacent to A as before: either by a swap of elements that are on the same row followed by a swapping of rows (if τ_{l^2+l+1} is in region C), or just by a swapping of rows (if τ_{l^2+l+1} is in region D). If τ_{l^2+l+1} is in region B, denote by r its column, and let τ_{max} be the maximum of the elements in regions C and D. By a swap of τ_{max} with an element at the intersection of one of the first l columns with the row of τ_{max}, we can bring τ_{max} in region D without increasing the cost. By swapping rows we can then bring τ_{max} on a cell adjacent to A without changing the cost.

Now we show that by an arbitrary swapping of the elements placed at the intersection of column r with the first l rows with the elements placed at the intersection of row $l+1$ with the first l columns, the cost of the placement g does not increase (see Figure 2). Note that, by this swap, τ_{l^2+l+1} arrives on a cell of row $l+1$ adjacent to region A, as required by our placement strategy.

To compare the costs of the two placements (before and after the swap), we only need to look at the cells for which the associated cost might change after this operation. These are the cells from column r below row $l+1$ as well as the cells from row $l+1$, situated to the right of column $l+1$, with the exception of cell $M_{l+1,r}$. The sum of their costs before the swap is:

$$(k-l-1) \cdot \tau_{l^2+l+1} + (k-l-2) \cdot \tau_{max}$$

while the sum of their costs after the swap is:

$$(k-l-1) \cdot \tau_{max} + (k-l-2) \cdot \tau_{l^2+l+1}$$

which is smaller than the first one, since $\tau_{max} \leq \tau_{l^2+l+1}$.

Thus we can bring τ_{l^2+l+1} on a cell of row $l+1$ adjacent to A without increasing the cost. By direct swaps we can bring the rest of the elements up to $\tau_{(l+1)^2}$ on positions $M_{l+1,2}, \ldots, M_{l+1,l+1}$ without increasing the cost. This completes the proof that g can be converted to a placement according to our strategy up to the first $(l+1)^2$ elements. By induction this shows that g can be completely converted to a placement according to our strategy without increasing the cost. Therefore our placement strategy is optimal. □