Clustered K-Center: Effective Replica Placement in Peer-to-Peer Systems

Jian Zhou
Univ. of California, Riverside
jianz@cs.ucr.edu

Xin Zhang
Carnegie Mellon Univ.
xzhang1@cs.cmu.edu

Laxmi Bhuyan
Univ. of California, Riverside
bhuyan@cs.ucr.edu

Bin Liu
Tsinghua Univ.
liub@mail.tsinghua.edu.cn

***Abstract*— Peer-to-Peer (P2P) systems provide decentralization, self-organization, scalability and failure-resilience, but suffer from high worst-case latencies. Researchers have proposed various replication algorithms to place multiple copies of objects across the network in pursuit of better performance for P2P computing; nevertheless, they neither presented clear analysis nor derived worst-case bound for their algorithms. In this paper, we model the replica placement problem arising in real-world P2P networks as a Clustered K-Center problem which we prove to be NP-complete. Then we propose an efficient approximation algorithm to this problem with a provable upper bound. Extensive experiments have been conducted to demonstrate the effectiveness and efficiency of our algorithm. The experimental results show that our approach can run several orders of magnitude faster than the optimal solution while being able to minimizing the query latency.**

Keywords: Replica Placement, Peer-to-Peer Network, Clustered K-Center Problem and Approximation Algorithm

I. Introduction

During the past few years, the emergence of Internet-scale distributed systems including storage administration in global companies, entertainment file sharing, and large distributed database systems has led to extensive research on efficient and scalable distributed computing architectures. Peer-to-Peer (P2P) computing, among many other distributed computing models, exhibit good scalability and stability. They have proved to be an efficient and successful way for distributed computing and file sharing over the Internet.

Recent Internet traffic measurements have shown that P2P traffic increases significantly and becomes one of the major traffic on the Internet. In some network segments, the P2P traffic even goes beyond the traditional web traffic [15]. Whereas, the P2P traffic distribution in terms of volume, connectivity and bandwidth among the peers are extremely skewed [20] where peers may easily get overloaded. When peers are overloaded, queries in the peers tend be dropped or be kept in a queue for a long time which degrades the system performance significantly in terms of success rate and client perceived latency.

Many works have shown that existing P2P protocols may not achieve a satisfactory worst-case client-perceived latency for large scale, latency sensitive applications [19]. Moreover, Peer-to-Peer system is known to be unstable as peers may join and depart the system arbitrarily. Once a peer gets failed or leaves its group without notifying others, it will result in query failures and the fault tolerance mechanisms in literature [17], [2] to get back the data incur lots of overhead to the system.

Replication algorithms are believed to be an effective method for improving the availability of data, enhancing performance of query latency and load balancing [1]. By distributing multiple copies of objects in the network and forwarding each query to its nearest copy, we can effectively reduce the query search latency and enhance the reliability of the system. Recently, a number of replication algorithms in P2P have been proposed to offload the workload as well as to avoid the network congestion [5], [16]. However, these replication strategies are of ad hoc paradigm and are lack of a clear analysis of the performance gain.

Replica placement algorithm is essentially a tradeoff between query latency and memory overhead (i.e. the overhead of storing the replications in the network). A good placement mechanism should demonstrate a maximum cost benefit given the memory space in accordance with an access pattern. This paper considers a static replication data placement problem aiming at minimizing the maximum query latency across the network. More specifically, given a network of nodes, communication cost function and individual storage capacity of each node, we study the problem of determining a placement of replication objects on the nodes such that the maximum query latency, taken over all nodes and all objects, is minimized. We name the optimization problem as Clustered K-Center problem as it bears some similarity to the classic k-center problem in graph theory. The k-center problem tries to find the set of k centers in an arbitrary graph such that the maximum shortest distance of all nodes to the nearest center is minimized. Our Clustered K-Center problem differs from the classic k-center problem in that instead of choosing a set of k nodes to place the replication and serve the others, it requires every node to serve as a center (server) as well as a client at the same time. In this paper, we prove that Clustered K-Center problem is NP-complete and develops an approximation algorithm to solve it in polynomial time.

The main contributions of this paper are described as follows,

- We formulate the optimal replica placement problem in P2P networks as a Clustered K-Center problem.
- We prove that the Clustered K-Center problem is NP-complete and design an approximation algorithm with quadratic time complexity.
- Comprehensive experiments have been conducted to

demonstrate the effectiveness and efficiency of our algorithm.

The rest of the paper is organized as follows. In Section II, we survey different replica placement protocols. Section III models the replica placement problem and proves the NP-completeness of the Clustered K-Center problem. Section IV presents our approximation algorithm in details. In section V, we analyze the time complexity of our algorithm and give a formal proof for the performance bound. Finally, experimental results are demonstrated in section VI and section VII concludes the paper.

II. Related Work

The replica placement problem has been studied extensively in many disparate fields, such as file assignment problem [7], file allocation [23], distributed databases [13], data management [14], etc. However, these fields require the replica placement algorithms to take into consideration the data writes, consistency, update propagation and compounded guarantees which are not major concerns in a read operation oriented P2P systems.

The majority works for replica placement in P2P systems are focused on content delivery networks (CDN [24], [10], [11]), where CDN nodes cooperate with each other to satisfy the requests made by end users. Usually, these problems have their own objective function with different constraints such as storage capacity, link capacity, node bandwidth capacity, etc. In [11], the authors try to minimize the average query latency with storage constraint. The replica placement problem is formulated as an integer linear programming problem and some heuristic algorithms are proposed. However, the author does not provide an analysis of the performance bound. In [24], a simple hierarchically placement algorithm is proposed to minimize the average access time with bandwidth constraints and the proof of the constant factor approximation is provided. Whereas the objective function is not the maximum client-perceived latency as we stress in this paper. In [10], the authors propose to place the tracers optimally such that the maximum distance from any client to its nearest tracer is minimized, but only part of the nodes are selected as tracers while others only act as clients which may deteriorate load balancing in P2P systems .

Generally, the problem of determining optimal replica placement in an arbitrary network is modeled as a classical graph theory problem, such as facility location, k-median and k-center. In the facility location problem, there is an open-up cost for replicating a copy at each node, and its goal is to minimize the sum of the open-up costs and the total query costs. While for the k-median and k-center problems, the objective is to minimize the total query latency and maximum latency respectively given the number of the replicas. All of the problems are NP-complete and [4], [12] propose the approximation algorithms. As the worst-case latency is usually the major concern in a QoS (quality of service) oriented network [22], the k-center problem is most closely related to our problem. However, the k-center problem is only suitable for putting the replicas for one service or one object (or one group of objects) on the selected k centers. Other peers in the network cannot contribute to the system as much as the k centers. This is the case for all the facility location, k-median and k-center problem. Only recently, Baev and Rajaraman [3] propose a 20.5-approximation algorithm for replica placement on all nodes in the network with storage constraint using a rounding technique of the integer linear programming. Their objective function is the average query latency. Our problem has not take into consideration the open-up cost of placing replicas since in a query intensive P2P system the open-up cost for placing the replicas is typically orders of magnitude smaller than the cost generated by queries.

III. Problem Formulation

We consider a set of n peers in a distributed peer-to-peer network, each peer stores various objects like video files, web pages or other arbitrary documents. Queries requesting for a specific object may originate from any node at any time within the network and they are all forwarded to the host node [1] where the object is placed. Here, the latency of a query is denoted by the total distance of the route from the querying node to the host node.

In order to reduce the query latency, researchers propose to replicate the objects on other nodes in the network. The problem becomes, given the replication number k, how to find the best k replication nodes, denoted as $R = r_1, r_2, ..., r_k$, such that the maximum latency to fetch the nearest copy of object is minimized. Here, we assume the system is able to forward the queries to the nearest replication r_i using existing mechanisms [21]. As such an effect, the network can actually be viewed as a forest of k search trees with each tree rooted at one replication node r_i. Thus all the k roots cooperatively serve the queries for the objects, and in this way the workload is shared among them. This can be modeled as a well-known graph problem: k-center problem which is described as following. Given a graph $G = (V, E)$ representing an overlay network topology where V denotes the set of nodes and E denotes the connections between the nodes, and given an integer k, compute a subset of k vertices $R \subseteq V$, such that the maximum distance between any vertex $v \in V$ and its nearest center $r_i \in R$ is minimized. The k-center problem is known to be NP-complete [12], and there exist several constant-factor approximation algorithms in the literature [9].

However, our problem is different. Consider a P2P network with thousands of objects and all of them are replicated on the same k centers, the k centers may easily get overloaded by the huge amount of queries. Moreover, the memory capacities of the k centers will be quickly saturated as the replications increase. The whole P2P systems becomes extremely load unbalanced. In other words, it is not a good idea to use the same k centers for all the objects when replicating the objects. For load balance purpose, nodes are required to keep roughly

[1]Host node for object o is a node which contains o and is ready to share with others.

the same number of objects across the network which may help to ensure the queries are evenly distributed among the nodes. Thus in our replication system, objects are grouped together and each group of objects can be placed on one set of k centers. Essentially, the network is partitioned into m^2 non-overlapping groups ($g_1, g_2, ..., g_m$) of k centers where $V = g_1 \cup g_2 \cup ... \cup g_m$ and each group serves for a group of objects, here we assume n can be divided by k integrally and $m = n/k$. For any $g_i (i \in [1, m])$, there is a cover radius d_i which is the maximum distance between any vertex $v \in V$ and its nearest center $r \in g_i$. Our problem now is given a graph $G = (V, E)$ with n nodes, and given an integer k, compute m non-overlapping groups such that the maximum cover radius among all the groups is minimized. From the graph coloring view, the problem can also be described as follows: given a graph with n nodes and a parameter k, color the vertices with m colors so that each color class has k vertices. The objective is to find the small value D such that the vertices within distance D of any vertex in the network contain at least one vertex of every color.

Theorem 3.1: The decision problem of *Clustered K-Center* problem is NP-complete.

Proof: In order to prove that our optimization problem is NP-complete, we first formulate the problem as a decision problem. Given an arbitrary undirected graph $G = (V, E)$ and k, let $D(i, j)$ denote the distance between node i and j where $i, j \in V$. Given a target distance T, we ask if there is a clustering $g_1, g_2, ..., g_m$ such that

$$\max_i \max_{v \in V} \min_{r \in g_i} D(v, r) \leq T \tag{1}$$

We prove the NP-completeness of this problem by showing that it belongs in NP and then we reduce the domatic number problem to a special case of our problem. This proves the NP-completeness.

The problem is easily seen to be in NP. Given a clustering $g_1, g_2, ..., g_m$ and the number of hops T, we can verify in polynomial time whether the worst case latency (formulated in equation 1) is less than T hops.

Next, we take a graph $G = (V, E)$ where the weight of each edge c_{ij} is 1 if $(i, j) \in E$. We consider the special case where $T = 1$ and all the edges in E have weight 1 (That is $c_{ij} = 1 \quad \forall (i, j) \in E$). In this case, we try to partition the network into m disjoint groups $g_1, g_2, ..., g_m$ in a way such that for each group g_i, all the vertices in G are directly connected to any vertices within the group g_i. Let N_i denote the vertices directly connected to vertices in g_i, the following equation is true for the special case G.

$$N_i = \{v_i | \exists j \in g_i, (i, j) \in E\}, \quad \forall i, N_i \cup g_i = V \tag{2}$$

In other words, each group g_i is a dominating set of the graph G. Thus, our problem is identical to the well-known NP-complete domatic number problem [8]. Given that our Clustered K-Center problem belongs in NP problem and that the domatic number problem reduces to it, we prove that our replica placement problem is NP-complete. ■

[2]Number of groups of centers. $m = n/k$

IV. Approximation Algorithm

As shown in section III, the replica placement problem cannot be solved easily in a large scale P2P system as it is NP-complete. In this section, we present a polynomial algorithm for the Clustered K-Center problem which guarantees the worst case latency is no more than $m-1$ factor of the optimal solution.

Recall that $G = (V, E)$ represents the network topology where each vertex in V is a peer in the network and each edge in E represents a link between the connected two peers. The latency of the link is denoted by an edge weight c_{e_i} where $e_i \in E$. In our algorithm, we use $G_d = (V, E_d)$ to denote the distance graph of G where each pair of vertices in V are connected, so G_d is actually a clique. The weight of the edge between node v_i and v_j in G_d is the sum of the total edges' weights on the shortest path from v_i to v_j in G. One observation is any distance graph satisfies the triangle inequality $c_{ij} \leq c_{ik} + c_{kj}$ for all $i, j, k \in E$, refer to lemma 5.1 for detailed proof.

It is not hard to see that the value of the optimal solution in Clustered K-Center problem is actually one of the shortest path between two nodes in G. Therefore, for any arbitrary graph $G = (V, E)$, the value of the optimal solution for cluster k-center problem is one of the edge weights in the distance graph G_d. Herein, the first step is to construct a distance graph G of the topology by calculating the shortest distance between every pair of nodes in G. This can be done by running a Dijkstra algorithm [6]. Furthermore, we label the edges of E_d so that $c_{e_1} \leq c_{e_2} \leq ... \leq c_{e_l}$ where $l = \binom{n}{2}$.

Begin with a graph $G_0 = (V, E_0)$ where $E_0 = \{e_j | j \leq 0\}$, we interactively add one edge with the smallest weight from the remaining edges that have not been added to the graph in each round. That is to say, after the ith round of the procedure, the produced graph is $G_i = (V, E_i)$ where $E_i = \{e_j | j \leq i\}$. Every time, a test is made to check whether the current graph G_i contains an appropriate feasible subgraph for clusters of k centers. However, it is NP-complete to evaluate whether the graph is eligible for producing clusters of k centers or not. Our strategy is to relax the decision procedure a bit such that it can finish in polynomial time with a provable performance bound for the worst case latency.

For each iteration, the graph G_i is evaluated to see whether it contains k connected components each with size m. This can be solved easily by a DFS (Depth First Search) traversal of the graph which is an $O(v+e)$ algorithm (here, v and e represent the number of vertices and edges in the graph respectively). Once we get a graph G_i satisfying the above condition, by the definition of the power of graph[3], it is not hard for us to get clustered k centers in G_i^{m-1}. As in G_i, each vertex $v_i \in V$ is

[3]Here, the power of graph is defined as follows: given an arbitrary graph $G = (V, E)$ and t is a positive integer, let the tth power of G be $G^t = (V, E^t)$, where there is an edge (u, v) in G^t wherever there is a path from u to v with at most t edges in G.

Approximation Algorithm

Input Parameters:
Underlying Topology: $G = (V, E)$
Number of Duplications: k $\quad (k * m = n)$

Output Parameters:
m groups of centers $g_1, g_2, g_3 ... g_m$

begin

(1) Construct a distance graph $G_d = (V, E_d)$ from the network topology G by computing all pairs of shortest distance
(2) Sort the edges in E_d
$c_{e_1} \leq c_{e_2} \leq ... \leq c_{e_l}$, where $l = \binom{n}{2}$;
(3) Initialize $i = 0$;
(4) Loop
$i = i + 1$;
$G_i = (V, E_i)$ where $E_i = \{e_j | j \leq i\}$;
$Condition$: $G_i = \bigcup_{i=1}^{k} S_i \quad \forall i, |S_i| \% m = 0$
S_i is a connected component
if($Condition$ is false)
Repeat Loop;
else
$G_i^{m-1} = \bigcup_{i=1}^{k} C_i \quad \forall i, |C_i| = m$
$C_i \bigcap C_j = \emptyset \quad \forall i, j \in 1, 2, ..., k$
(5) Greedily select one node from each cluster C_i to form a group g_j
(6) Finish. Return $g_1, g_2, g_3 ... g_m$

end

Fig. 1. Approximation Algorithm for the Clustered K-Center Problem

within a connected component containing m vertices (denoted as V_i including v_i), every two vertices in any V_i need $m-1$ steps to reach each other. According to the definition of power of graph, every V_i is a clique with $|V_i|$ nodes in G_i^{m-1}. It is easy to see that G_i^{m-1} can be partitioned into k cliques each with m members. Now, we choose one vertex from each clique exclusively to form a group of k centers g_i. Thus, we cluster the network G into m groups where each group acts as k centers for part of the objects in the system. If the test fails, we repeat the procedure to add one more edge. Otherwise, we produce a power of $(m-1)$ of the current graph and form the m clusters out of it.

V. Analysis

We demonstrated how our algorithm works to cluster the nodes in the network to form the groups of k centers in section IV. In this section, we first analyze the time complexity of the algorithm and then show that the worst case latency produced by our algorithm is no more than $(m-1)$ factor of the optimal solution.

A. Time Complexity Analysis

In this subsection, we analyze the time complexity of our approximation algorithm and show that it can be finished within $O(n^4)$ where n is the number of nodes in the network.

At the very beginning, we construct a distance graph by calculating the shortest distance among all pairs of nodes in G which is a multiple source shortest path problem. To our knowledge, a single source multiple destination shortest path problem can be solved by a Dijkstra's algorithm with run time $O((n+e)logn)$ where n and e denote the number of nodes and edges respectively. When constructing the distance graph, it is n time of Dijkstra's algorithm which is $O(n(n+e)logn)$. Since the number of edges is at most $\binom{n}{2} = n(n-1)/2$, the time complexity for constructing the distance graph is $O(n^3 logn)$. When we label the edges in G_d in an ascending order, it is actually a sort algorithm for the $\binom{n}{2}$ edges. That is at most $O(n^2 logn)$.

We can further see that the number of the loop iterations cannot exceed $\binom{n}{2}$ times. During each iteration, we run a DFS (Depth First Search) to traverse the graph to check the eligibility of G_i which runs in $O(n+e)$. Thus, the time complexity for the loop is $O(n^2(n+e))$.

The time complexity of our algorithm is the sum of the three parts $O(n^3 logn) + O(n^2 logn^2) + O(n^2(n+e))$, which makes $O(n^4)$.

B. Approximation Ratio

In this subsection, we place a bound on the bottleneck distance in the clustered k centers produced by the algorithm. Let d_{opt} denote the optimal solution and d_G denote the worst case latency to the nearest center among all the m groups of centers, we prove in theorem 5.4 that $d_G \leq (m-1) * d_{opt}$ for any arbitrary graph. Here, **approximation ratio** is d_G / d_{opt}.

As far as we know, most of the network topologies do not necessarily follow the triangle inequality metric. For example, the direct link between node i and j might be very slow due to the network congestion while another path linking i and j might be relatively faster. However, the distance graph for any network satisfies the metric.

Lemma 5.1: Let $G = (V, E)$ be an arbitrary subgraph of a distance graph $G_d = (V, E_d)$, then G satisfies triangle inequality.

Proof: Suppose that there exists an edge $(i, j) \in E$ such that there is a node $k \in V, k \neq i \cap k \neq j$ which produces $c_{ij} > c_{ik} + c_{kj}$. This means there exists a shorter path from node i to node j than the existing direct edge in distance graph. This contradicts to the definition of the distance graph which has the length of the shortest path to represent the edge weight. ■

If $G = (V, E)$ is an arbitrary graph, let $max(G) = \max_{e_i \in E} c_{e_i}$. Our approximation technique is based on some nice properties of the power of graph which are demonstrated as the following Facts.

Fact 5.2: Let G be any subgraph of a distance graph, then $\max(G^t) \leq t * \max(G)$.

Proof: According to the definition of power graph, a path with no more than t edges in G produces an edge in G^t. Since the weight of each edge on the path in G is at most $max(G)$, the sum of all the edges' weights on the path is no more than $t * max(G)$. By the triangle inequality, the weight of any edge (i, j) in G^t is no more than that of the entire path from i to j. Therefore the weight of (i, j) is at most $t * max(G)$. ■

For the Clustered K-Center problem, there is a set of feasible subgraphs S of distance graph G_d each element of which has non-overlapping m groups of k centers. Among all of the

subgraphs, the optimal solution tries to find the best subgraph G in S with minimized $max(G)$. In the algorithm presented above, we test in each iteration whether the G_i^{m-1} contains a subgraph $G' \in S$.

Lemma 5.3: If $G = (V, E)$ contains a feasible subgraph in S for Clustered K-Center problem, every vertex in V must have degree more than $m - 1$.

Proof: Let $g_1, g_2, ..., g_m$ denote the groups of the k centers as a feasible solution for the Clustered K-Center problem. By the definition of K-Center , every vertex v in V is connected to its nearest center. Thus, unless v is a center itself, it will be connected to one center in $g_i, i = [1, m]$. That is to say, for any vertex $v \in g_i$, it is connected to at least one node in each group except the one it belongs to. Thus, any vertex must be connected to $m - 1$ nodes in G. ■

Theorem 5.4: For any network topology $G = (V, E)$ with triangle inequality metric, $d_G \leq (m - 1) * d_{opt}$.

Proof: Suppose the procedure in Fig.1 terminates at round s where the graph is G_s. In order to prove this result, we show that any graph G_i ($i < s$) before the termination does not contain a subgraph which is a feasible solution to the Clustered K-Center problem while G_s^{m-1} contains a feasible solution.

Now we prove that G_i does not contain a feasible solution to the problem. Suppose that there is a feasible solution for some G_i, then according to lemma 5.3, every vertex in G_i must have direct neighbors more than $m-1$. That means any vertex is in a connected component with size at least m. However, any G_i produced by the procedure at least has one vertex not contained in a connected component. This contradiction proves that any graph before G_s cannot be grouped into m groups of k centers. This actually indicates that

$$\forall i \leq s \quad c_{e_i} \leq d_{opt} \tag{3}$$

Next, we show that G_s^{m-1} contains a subgraph of the feasible solution. Let $V_1, V_2, ..., V_k$ denote the disjoint connected components in G_s. From the procedure, we know that the size of V_i is ensured to be m. By the definition of the power of graph, the graph induced on V_i in G_s^{m-1} is a clique on $|V_i|$ vertices as every vertex in any V_i needs $m - 1$ edges to reach each other. Now, we choose one vertex from each clique to form a group of centers exclusively, thus we get non-overlapping groups $g_1, g_2, g_3, ..., g_m$ where each group is a dominant set of G_s. The worst case latency in our solution d_G is actually one of the edge in G_s^{m-1}. The longest edge in G_s is c_{e_s}. By Fact 5.2, the graph G_s^{m-1} has edges no longer than $(m - 1) * c_{e_s}$. From equation 3, we get the following result.

$$d_G \leq (m - 1) * c_{e_s} \leq (m - 1) * d_{opt}$$ ■

VI. Performance Evaluation

In this section, we evaluate the performance of our approximation algorithm for the Clustered K-Center problem. Extensive experiments are conducted under various network topologies and a wide range of network size.

In the experiments, the network topology is generated with a GT-ITM generator [25] which can efficiently generate Transit Stub models that accurately reflect the topological properties of the real internet. TS models the networks using a two-level hierarchy of routing domains, with transit domains interconnecting the lower level stub domains. By default, the latency of intra-transit domain links, stub-transit links and intra-stub domain links are set to 20ms, 5ms and 2ms respectively[18].

To highlight the effectiveness and efficiency of our algorithm, we compare our algorithm against two other approaches, the optimal approach and a natural greedy approach. In the optimal approach, we identify the optimal solution using a brute force search which systematically enumerates all possible candidates for the solution. As we showed in sectionIII, the Cluster K-Center problem is a NP-complete problem which indicates that it cannot be solved in polynomial time. Therefore, the optimal solution can only be achieved by evaluating all the possible configurations whose time complexity is exponential to the network size n. With the greedy approach, the m clusters of centers are identifies one by one. For the first cluster of k centers, we compute the best k centers using a greedy approximation algorithm for the k-center problem [9] and remove the cluster from the original graph. In the remaining graph, the second cluster is identifies with the same mechanism and vice versa. The time complexity of the greedy approach is $O(n^3 logn)$.

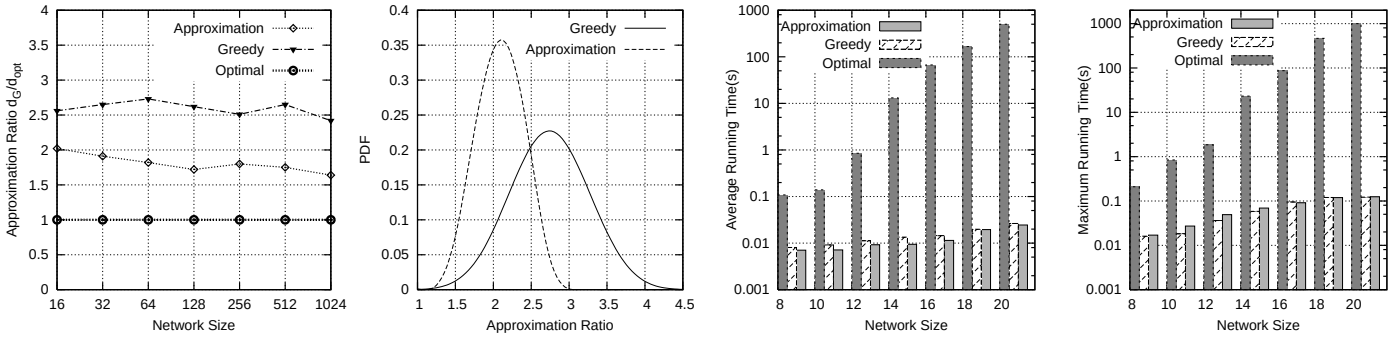
Approximation algorithm is an approach to attacking difficult optimization problems efficiently. The approximation ratio and time complexity are two important metrics to evaluate the performance of approximation algorithms. In the following subsections, we study the performance of our algorithm against various network topologies and compare the approximation ratio and time complexity of various approaches.

A. Approximation Ratio

In this subsection, we examine how well our algorithm approximate the optimal solution of the Clustered K-Center problem by depicting the approximation ratio of our algorithm.

The Transit-Stub network topology is used here since it is proven to be correlate well with the internet structure. The number of duplications of objects are automatically increased as the network size increase to achieve the target performance. For easy comparison among different schemes, we set the number of duplications k to be $n/4$ in our experiment. That is to say, the number of groups m is four by default. As we proof in section III, Clustered K-Center problem is a NP-complete problem which cannot be solved in polynomial time. In other words, it is very computation intensive even for a small scale topology. For comparison, we compute a tight lower bound for the optimal solution for any graph with $O(n^2)$ time complexity.

As shown in Fig.2(a), our algorithm approximates the optimal solution with ratio less than three consistently. This verifies our analysis in section V which claims that our approach is a $m - 1$ approximation algorithm with arbitrary



(a) Approximation Ratio (b) PDF of Approximation Ratio (c) Average Running Time (d) Worst Case Running Time

Fig. 2. Performance Analysis

network topology. With the varying network size, our algorithm outperforms the natural greedy approach consistently. In Fig. 2(b), we depict the probability distribution function (PDF) of the approximation ratio for our algorithm and the greedy algorithm. Both of the greedy approach and our approximation algorithm follow the normal distribution. However, the greedy approach shows a heavy tail which indicates that it performs badly in some particular cases. While approximation ratio of our algorithm demonstrates a normal distribution with the worst case within $m-1$ and the average case around $\frac{m-1}{2}$.

B. Running Time

In this subsection, we compare the average and the worst case running time of the Clustered K-Center problem using our approximation algorithm, the greedy heuristic and brute force search. From the previous analysis, we know that the time complexity of brute force search increase exponentially with the network size while our algorithm has quadratic time complexity. This trend can be clearly captured in Fig. 2(c) and 2(d) where our algorithm and the greedy approach is several orders of magnitude faster than the brute force search algorithm. Although the greedy heuristic is faster than our approximation approach in the worst case, our algorithm outperforms the greedy heuristic in the average case.

From the experiments, we demonstrate that our approximation algorithm can approximate the optimal solution consistently well and is several orders of magnitude faster than the brute force search.

VII. Conclusion

In this paper, we show that the replica placement problem in P2P networks can be represented as a new Clustered K-Center problem (which essentially differs from the classic k-center problem) and is proven to be NP-complete. To solve this problem, we bring forward an approximation algorithm in the form of a distance graph for the network topology; when our defined feasibility condition holds at a certain point, the replica placement solution can be built out of (m-1) power of current distance graph. Theoretical analysis shows that our algorithm incurs $O(n^4)$ time complexity and the worst-case latency yielded by our algorithm is no more than (m-1)-factor of the optimal solution.

References

[1] S. Androutsellis-Theotokis, D. Spinellis, "A survey of peer-to-peer content distribution technologies", *ACM Comput*, 2004.

[2] J. Aspnes, Z. Diamadi, and G. Shah, "Fault-tolerant routing in peer-to-peer systems", *PODC*, July 2002.

[3] I. Baev and R. Rajaraman, "Approximation algorithms for data placement in arbitrary networks", *In SODA*, 2001.

[4] M. Charikar and S. Guha, "Improved combinatorial algorithms for the facility location and k-median problems",*In FOCS*, 1999.

[5] Y. Chen, R. Katz and J. Kubiatowicz, "Dynamic Replica Placement for Scalable Content Delivery", *IPTPS*, 2002

[6] E. W. Dijkstra, "A note on two problems in connetion with graphs", *Numerische Mathematik* , 1959.

[7] L.W. Dowdy and D. V. Foster, "Comparative Models of the File Assignment Problem," *ACM Computer Surveys, 14(2), pp. 287C313*, 1982.

[8] U. Feige, M. M. Halldorsson, and G. Kortsarz, "Approximating the domatic number", *32nd Ann. ACM Symp. on Theory of Computing*, 2000

[9] T. F. Gonzalez, "Clustering to Minimize the Maximum Intercluster Distance", *Theoretical Computer Science*, June 1985.

[10] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "On the Placement of Internet Instrumentation", *IEEE INFOCOM*, March 2000.

[11] J. Kangasharju, J. Roberts, and K. W. Ross, "Object Replication Strategies in Content Distribution Networks", *WCW'01*, June 2001.

[12] O. Kariv and S. Hakimi, "An algorithm approach to network location problems. I. the *p-center*", *SIAM Journal of Applied Mathmatics 37*, 1979.

[13] D. Levin and H. Morgan, "Optimizing Distributed Data Bases C A Framework for Research," *AFIPS*, 1975.

[14] B. Maggs, F. Meyer auf der Heide, B. Vocking, and M. Westermann, "Exploiting Locality for Data Management in Systems of Limited Bandwidth", *Symp. on Foundations of Computer Science*, Oct. 1997.

[15] M. Meeker, "The state of the Internet", *2006 Web 2.0 Summit*, Nov. 2006, San Francisco, USA.

[16] C. G. Plaxton, R. Rajaraman, A. W. Richa, "Accessing Nearby Copies of Replicated Objects in a Distributed Environment", *SPAA*, 1997.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network", *ACM SIGCOMM*, Aug. 2001.

[18] S. Ratnasamy, M. Handley, R. Karp and S. Shenker, "Topologically Aware Overlay Construction and Sever Selection", *IEEE Infocom*, 2002.

[19] S. Rhea, B. Chun, J. Kubiatowicz, S. Shenker, "Fixing the Embarassing Slowness of OpenDHT on PlanetLab", *Proceedings of the Second Workshop on Real, large Distributed System (WORLDS' 05)*, 2005.

[20] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic Across Large Networks", *ACM/IEEE Transactions on Networking, 12(2)*, April 2004.

[21] X. Tang and J. Xu, "On replica placement for QoS-aware content distribution", *In the Proc. of INFOCOM*, 2004.

[22] L. Shu and M. Young, "Real-time concurrency control with analytic worst-case latency guarantees", *In Proc. of the 10th IEEE Workshop on Real-Time Operating Systems and Software*, N.Y., May 1993.

[23] R. Tewari and N. Adam, "Distributed File Allocation with Consistency Constraints," *ICDCS*, 1992, pp. 408C415.

[24] A. Venkataramanj, P. Weidmann, and M. Dahlin, "Bandwidth Constrained Placement in a WAN," *PODC*, August 2001.

[25] E. Zegura, K. Calvert and S. Bhattacharjee, "How to Model an Internetwork", *Proceedings of IEEE Infocom*, CA, MAY 1996.