On Server Provisioning for Distributed Interactive Applications

Hanying Zheng and Xueyan Tang
School of Computer Engineering
Nanyang Technological University
Singapore 639798
Email: {zhen0084, asxytang}@ntu.edu.sg

***Abstract*—Increasing geographical spreads of modern distributed interactive applications (DIAs) make distributed server deployment vital for combating network latency and improving the interactivity among participants. In this paper, we investigate the server provisioning problem that concerns where to place servers in DIAs. We formulate the server provisioning problem with an objective of reducing the network latency involved in the interaction between participants. We prove that the problem is NP-hard under *any* one of the following three scenarios that may be common in practice: (a) the network latency does not satisfy the triangle inequality; or (b) the choices of server locations in the network are restricted; or (c) the number of server locations to select is limited. Then, we propose an efficient greedy server provisioning heuristic, analyze its approximation ratio and give a tight example. Experiments using real Internet latency data show that our proposed algorithm significantly outperforms traditional k-median and k-center server placements.**

I. Introduction

Benefiting from the rapid growth of Internet technologies, an increasing number of Distributed Interactive Applications (DIAs) are emerging to provide people with new ways of collaboration and entertainment. In these applications, participants dispersed at different locations interact with each other through the network in real time. Examples of DIAs include online gaming [1], instant messaging [2], collaborative computer-aided design and engineering [3], and web-based e-learning [4].

A critical Quality of Service measure in DIAs is the time lag experienced by the participants during their interaction. The interaction is usually carried out in the form of updates to the application state (such as the virtual game worlds in online gaming and the shared workspaces in collaborative design tools). The interaction process normally involves communicating user-initiated operations and their resultant state updates between the participants and the servers as well as executing the operations at the servers. Thus, the time lag in the interaction includes the network latency in the communication and the processing delay at the servers. The latter is generally easier to cut than the former. In particular, the emerging cloud computing paradigm enables customers to rent computing resources purely on demand for hosting their applications [5]. Although elastic computing power supply from the clouds can largely minimize the server-side processing delay in DIAs, the network latency remains as a major barrier to achieving high quality interaction experience. Excessive network latency can severely degrade the participant's quality of experience in DIAs [1].

Increasing geographical spreads of participants in modern DIAs necessitate distributed server deployment to combat network latency [6]. In this paper, we focus on reducing the network latency involved in the interaction between participants by considering where to place servers in the network. We formulate the server provisioning problem for DIAs as a combinatorial optimization problem and prove that it is NP-hard under *any* one of the following three scenarios that may be common in practice: (a) the network latency does not satisfy the triangle inequality; or (b) the choices of server locations in the network are restricted; or (c) the number of server locations to select is limited. We then propose an efficient greedy heuristic for server provisioning, and conduct a theoretical analysis of its approximation ratio. The proposed algorithm is also experimentally evaluated with real Internet latency data. The results show that our algorithm significantly outperforms traditional k-median and k-center server placements.

Related Work. The classical k-median and k-center problems are widely used to model the placement of web server replicas in the Internet [7]–[9]. The k-median placement aims to place a given number of k servers to minimize the total distance (latency) from the clients to their nearest servers, whereas the k-center placement aims to place k servers to minimize the maximum distance (latency) from the clients to their nearest servers. Both problems are NP-hard [10]. A variety of heuristic approaches have been explored for these problems [9], [11]–[13].

The k-median and k-center server placements are, however, not suited to DIAs because DIAs are fundamentally different from web content delivery. The clients in the web just download contents from web servers. Thus, their access performance can be optimized by simply minimizing the client-to-server latency. In contrast, the clients in DIAs are engaged in mutual interactions among themselves. Each client connects to one server through which it interacts with all the other clients. Therefore, the interaction time between clients must include not only the network latencies from the clients to their connected servers but also the latencies between their connected servers. This renders straightforward k-median or k-center formulations invalid for server provisioning in DIAs. As shall be shown by our experiments, aggressively reducing the client-to-server latency alone may considerably increase the latency between the servers and thus make the interactivity between clients far worse than optimum. From the computability perspective, the mutual interaction feature also makes server provisioning for DIAs much more challenging than that for web content delivery. With the client-to-server latency as the sole optimization objective, placing more servers in the network can only improve the web access performance. Nevertheless, deploying more servers in DIAs does not necessarily imply shorter network latency in the interaction between clients. We show in this paper that the server provisioning problem for DIAs is NP-hard even if there is no limit on the number of servers that can be placed in the network. In a recent work, we have investigated client assignment strategies for enhancing the interactivity performance of DIAs given a set of servers placed [14]. This paper complements our earlier work by studying server provisioning for DIAs.

The rest of this paper is organized as follows. Section II formulates the server provisioning problem and Section III analyzes its hardness. Section IV presents the greedy server provisioning heuristic and the theoretical analysis of its approximation ratio. Experimental evaluations are elaborated in Section V. Finally, Section VI concludes the paper.

II. Problem Formulation

We model the network infrastructure underlying the DIA as a graph $G = (V, E)$, where V is the set of nodes and E is the set of links. For each pair of nodes $(u, v) \in V \times V$, we denote by $d(u, v)$ the latency of the network path between nodes u and v. We define $d(v, v) = 0$ for each node $v \in V$.

Without loss of generality, we assume that servers can be placed only at a particular set of nodes $Z \subseteq V$ in the network and refer to these nodes as *candidate server locations*. Z can be defined in different ways to address different server provisioning scenarios. For example, there are two typical types of models for running DIAs. In the traditional client-server model, the application state is maintained by dedicated servers. Participants, known as clients, are responsible for sending user-initiated operations to the servers for execution and receiving the state updates from the servers. In this case, the candidate server locations (e.g., the data centers operated by cloud providers) are normally separate from client locations. In the peer-to-peer model, on the other hand, the clients are responsible for executing operations and maintaining the application state by themselves. In this case, to select which clients to take up the role of operation execution, the candidate server locations Z can be modeled as the set of clients.

Let $C \subseteq V$ be the set of clients in the network. To participate in the DIA, each client $c_i \in C$ needs to connect to one server. The clients are often autonomous in deciding which servers to connect to. An intuitive and widely used strategy in many applications is for the clients to connect to their nearest servers, i.e., the servers with the shortest network latency to them [14]–[17]. Suppose that a set of locations $S \subseteq Z$ are selected to place servers. For each client $c_i \in C$, we denote by $n(c_i, S)$ the nearest server to c_i, i.e., $d(c_i, n(c_i, S)) = \min_{s \in S} d(c_i, s)$. Then, all the operations issued by client c_i would be sent to $n(c_i, S)$.

The interaction between two clients c_i and c_j goes through their connected servers. On receiving an operation issued by c_i, its server $n(c_i, S)$ forwards the operation to c_j's connected server $n(c_j, S)$ if they are different. Then, $n(c_j, S)$ executes the operation[1] and delivers the resultant state update to c_j to present the effect of c_i's operation. Thus, the interaction process involves the paths from c_i to $n(c_i, S)$, from $n(c_i, S)$ to $n(c_j, S)$, and from $n(c_j, S)$ to c_j. We refer to the concatenation of these paths as the *interaction path* from c_i to c_j. The length of the interaction path is given by $d(c_i, n(c_i, S)) + d(n(c_i, S), n(c_j, S)) + d(n(c_j, S), c_j)$ and represents the network latency involved in the interaction between c_i and c_j. Note that the interaction path from a client c_i to itself is the round trip between c_i and its connected server $n(c_i, S)$, whose length indicates the network latency involved for c_i to see the effect of its own operation.

We measure the interactivity performance of the DIA by the average interaction path length between all client pairs:

$$\frac{1}{|C|^2} \sum_{c_i \in C} \sum_{c_j \in C} \Big(d(c_i, n(c_i, S)) + d(n(c_i, S), n(c_j, S)) + d(n(c_j, S), c_j) \Big).$$

The shorter the average interaction path length, the higher the interactivity performance of the DIA. Since the total number of client pairs is fixed given the client set, to minimize the average interaction path length, it is equivalent to minimize the total interaction path length between all client pairs. Therefore, the server provisioning problem for DIAs is defined as follows.

Server Provisioning Problem. *Given a set of clients C and a set of candidate server locations Z in the network, select a set of locations $S \subseteq Z$ to place servers for the DIA that minimizes the total interaction path length between all client pairs, i.e.,*

$$\textit{minimize} \sum_{c_i \in C} \sum_{c_j \in C} \Big(d(c_i, n(c_i, S)) + d(n(c_i, S), n(c_j, S)) + d(n(c_j, S), c_j) \Big),$$

where $n(c_i, S)$ refers to the nearest server in S to client c_i.

[1]In some applications such as instant messaging, operation execution is as simple as forwarding the message typed by the user. In other applications like online gaming, operation execution could involve more complex computation and modification to the application state.

III. Hardness Analysis

The server provisioning problem is trivial if (1) the network latencies among the nodes satisfy the triangle inequality; (2) all the nodes in the network are candidate server locations; and (3) there is no limit on the number of server locations to select (or more precisely, the number of server locations selected can be as large as the number of clients because there is certainly no need to place more servers than the number of clients).

The triangle inequality implies that for any two clients c_i and c_j, and any two servers s_a and s_b, we have $d(c_i, s_a) + d(s_a, s_b) + d(s_b, c_j) \geq d(c_i, c_j)$. Therefore, under the above three assumptions, the optimal server provisioning solution is to place servers at all the nodes where the clients are located. In this way, each client connects to the server co-located with it so that the latencies between all clients and their nearest servers are 0. Thus, the interaction path length between any two clients c_i and c_j is simply $d(c_i, c_j)$, which is the shortest possible.

Interestingly, if *any* one of the above assumptions is relaxed, the server provisioning problem becomes NP-hard.

A. Networks without Triangle Inequality

If the network latency does not satisfy the triangle inequality,[2] we can show the NP-hardness of the server provisioning problem by a polynomial reduction from the minimum set cover problem [10]. Given a finite set P and a collection $\mathbb{Q}$ of its subsets, and a positive integer $k \leq |\mathbb{Q}|$, the decision version of the minimum set cover problem is to find out whether $\mathbb{Q}$ contains a subcollection $\mathbb{Q}'$ of at most k subsets such that $\bigcup_{Q \in \mathbb{Q}'} Q = P$.

Consider an instance $\mathcal{R}$ of the minimum set cover problem. Suppose that set P contains n elements: $P = \{p_1, p_2, ..., p_n\}$, and collection $\mathbb{Q}$ contains m subsets: $\mathbb{Q} = \{Q_1, Q_2, ..., Q_m\}$. We construct a network $G_{\mathcal{R}}$ consisting of the node set $V_C \cup V_S$, where V_C contains n nodes $V_C = \{c_1, c_2, ..., c_n\}$, and V_S contains k groups of nodes $V_S = \bigcup_{i=1}^{k} V^i$. Each node $c_i \in V_C$ corresponds to an element p_i in set P. Each group V^i contains m clusters of nodes $V^i = \bigcup_{j=1}^{m} V^i_j$. Each cluster V^i_j corresponds to a subset Q_j, and contains $|Q_j|$ nodes that correspond to the elements of Q_j. Thus, there are a total of $k \cdot \sum_{j=1}^{m} |Q_j|$ nodes in V_S.

Among the nodes in V_S, the latency between any two nodes in the same cluster or in different groups is set to 1. The latency between any two nodes in different clusters of the same group is set to L, where $L > 3n^2$. The latency between any two nodes in V_C is also set to L. The latency from a node $c_i \in V_C$ to a node $v \in V_S$ is set to 1 if v corresponds to element $p_i \in P$, and is set to L otherwise. Figure 1 illustrates an example of network $G_{\mathcal{R}}$, where each node pair connected by a link has latency 1, and each pair not connected by a link has latency L.

[2]This is not uncommon as Internet routing is often based on business policies and is not optimal in terms of network latency [18].

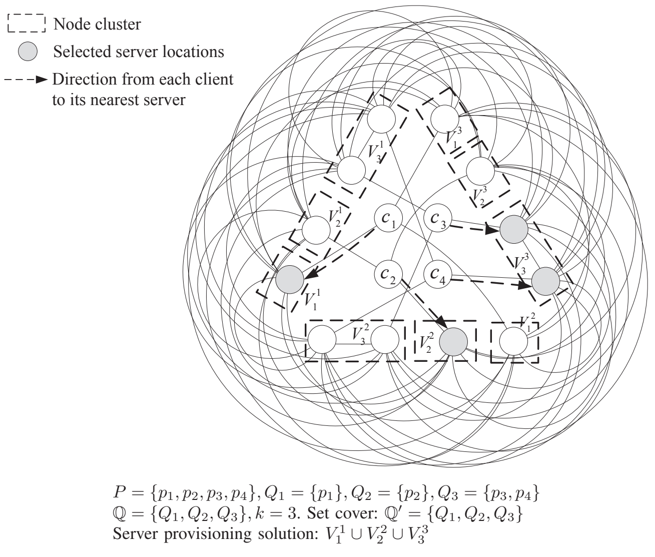
Node cluster
● Selected server locations
Direction from each client to its nearest server
V_3^1
V_1^3
V_2^3
V_2^1
c_1
c_3
c_2
c_4
V_3^3
V_1^1
V_3^2
V_2^2
V_1^2
$P = \{p_1, p_2, p_3, p_4\}, Q_1 = \{p_1\}, Q_2 = \{p_2\}, Q_3 = \{p_3, p_4\}$
$\mathbb{Q} = \{Q_1, Q_2, Q_3\}, k = 3$. Set cover: $\mathbb{Q}' = \{Q_1, Q_2, Q_3\}$
Server provisioning solution: $V_1^1 \cup V_2^2 \cup V_3^3$

Figure 1. Example instances of the minimum set cover problem and the server provisioning problem

An instance $\mathcal{T}$ of the server provisioning problem in the decision version is then defined on the constructed network $G_{\mathcal{R}}$ as follows: Suppose that a client is located at each node in V_C, all the nodes in the network are candidate server locations, and there is no limit on the number of server locations to select. Can we select a set of server locations such that the total interaction path length between all client pairs is bounded by $B = 3n^2$?

We first prove that if $\mathbb{Q}$ contains a set cover of size at most k for instance $\mathcal{R}$, there must exist a server placement with total interaction path length bounded by B for instance $\mathcal{T}$. Let $\mathbb{Q}' = \{Q_{x_1}, Q_{x_2}, \cdots, Q_{x_l}\}$ (where $1 \leq l \leq k$) be a set cover of size not exceeding k for instance $\mathcal{R}$. Then, placing servers at all the nodes in clusters $V^1_{x_1}, V^2_{x_2}, \cdots, V^l_{x_l}$ is a valid server provisioning solution for instance $\mathcal{T}$. In fact, since $\mathbb{Q}'$ is a set cover, each element $p_i \in P$ is contained in at least one subset among Q_{x_1}, Q_{x_2}, $\cdots$, Q_{x_l}. Thus, for each client $c_i \in V_C$, there exists at least one node in $V^1_{x_1} \cup V^2_{x_2} \cup \cdots \cup V^l_{x_l}$ having latency 1 to c_i. As a result, the latencies between all the clients and their nearest servers are 1. In addition, the latency between any two servers in $V^1_{x_1} \cup V^2_{x_2} \cup \cdots \cup V^l_{x_l}$ is 1 because they either come from the same cluster or from different groups. Therefore, the interaction path length between each pair of clients is bounded by $1 + 1 + 1 = 3$. Since there are n^2 interaction paths in total, the total interaction path length is bounded by $3n^2 = B$.

Next, we prove that if a valid server placement solution can be found for instance $\mathcal{T}$, there must exist a set cover of size at most k for instance $\mathcal{R}$. Let $S \subseteq V_C \cup V_S$ be the set of server locations selected in a valid solution of instance $\mathcal{T}$. Without loss of generality, we assume that the server at each location in S is connected by at least one client.[3] Since each client connects to its nearest server, if a server is placed at a node in V_C, the client at that node should connect to this server. Note that the latency between any two nodes in V_C is L. If servers are placed at more than one node in V_C, the interaction path length between the clients at these nodes becomes $L > 3n^2 = B$. Therefore, at most one node in V_C can be selected to place a server, i.e., $|S \cap V_C| \leq 1$.

If $|S \cap V_C| = 1$, there is exactly one server located in V_C and the remaining servers are all located in V_S. Let the server in V_C be located at node c_x. Since the latency between any two nodes in V_C is L, to bound the total interaction path length by $B < L$, the clients in $V_C \setminus \{c_x\}$ should all connect to the servers in V_S. Moreover, each of these clients must have latency 1 to its nearest server because the latency from a client to a server in V_S is either 1 or L. Thus, the clients in $V_C \setminus \{c_x\}$ must connect to servers not corresponding to element $p_x \in P$. Therefore, the latencies from c_x to these servers are L, exceeding the bound B. So, there is no valid solution of instance $\mathcal{T}$ satisfying $|S \cap V_C| = 1$.

If $|S \cap V_C| = 0$, all the servers are placed in V_S. Similar to the former case, since the total interaction path length does not exceed B, each client c_i must have latency 1 to its nearest server. This implies that each element $p_i \in P$ is covered by the subset Q_j corresponding to the node cluster of c_i's nearest server. Thus, $\mathbb{Q}' = \{Q_j \mid \exists i, V_j^i \cap S \neq \emptyset\}$ is a set cover for P. Furthermore, to cap the total interaction path length by B, all the servers must have latency 1 from each other. Thus, for each group of nodes V^i, all the servers in $V^i \cap S$ must come from the same cluster of V^i. Since there are k groups of nodes $V^1, V^2, \cdots, V^k$ in V_S, we have $|\mathbb{Q}'| \leq k$. Therefore, $\mathbb{Q}'$ is a set cover of size at most k.

Hence, a set cover of size at most k can be found for instance $\mathcal{R}$ if and only if there exists a server placement with total interaction path length bounded by B for instance $\mathcal{T}$. Thus, the server provisioning problem for networks without the triangle inequality is NP-hard.

The above hardness analysis also leads to the following non-approximability result.

Theorem 1. *For networks without the triangle inequality, the server provisioning problem cannot be approximated within a factor of any polynomial-time computable function of n unless P = NP, where n is the number of clients.*

Proof: For any instance of the minimum set cover problem, if there exists a set cover of size at most k, the total interaction path length of an optimal server placement must be bounded by $B = 3n^2$ in the network constructed above. Any non-optimal server placement, however, produces a total interaction path length greater than L. This is because in a non-optimal placement, either a client connects to a server that has a latency L to it, or the latency between a certain pair of servers is L. In either case, there is at least one interaction path longer than L, so the total interaction path length exceeds L. Similarly, if there does not exist any set cover of size at most k in the set cover problem, the total interaction path length of an optimal server placement in the constructed network is also greater than L.

Assume on the contrary that there exists a polynomial-time server provisioning algorithm with an approximation ratio of a polynomial-time computable function $\alpha(n)$. For any instance of the set cover problem, we set $L = \alpha(n)B = 3n^2\alpha(n)$ in the constructed network and then run the $\alpha(n)$-approximation algorithm. By checking whether the output server placement produces a total interaction path length within L, we can decide whether the optimal server placement would have a total interaction path length within B and thus whether there exists a set cover of size at most k. This implies a polynomial-time algorithm for the set cover problem, which contradicts to P $\neq$ NP.

Hence, the theorem is proven. ∎

B. Restricted Choices of Server Locations

If not all the nodes in the network are candidate server locations, the server provisioning problem becomes NP-hard as well. We can again prove it by a polynomial reduction from the minimum set cover problem.

Given an instance $\mathcal{R}$ of the minimum set cover problem, we construct the same network $G_{\mathcal{R}}$ as in the previous section, except that the node pairs that had latency L earlier have latency 2 now to satisfy the triangle inequality. That is, in the illustration of Figure 1, a node pair has latency 1 if they are connected by a link and has latency 2 otherwise. Suppose that a client is located at each node in V_C, only the nodes in V_S are candidate server locations, and there is no limit on the number of server locations to select. An instance $\mathcal{Y}$ of the server provisioning problem is then defined on network $G_{\mathcal{R}}$ by setting a bound $H = 3n^2 - n$ on the total interaction path length between all client pairs.

We first show that a set cover of size at most k for instance $\mathcal{R}$ gives rise to a valid server provisioning solution for instance $\mathcal{Y}$. Again, let $\mathbb{Q}' = \{Q_{x_1}, Q_{x_2}, \cdots, Q_{x_l}\}$ (where $1 \leq l \leq k$) be a set cover of size not exceeding k. Then, a valid solution for instance $\mathcal{Y}$ is to place servers at all the nodes in clusters $V_{x_1}^1, V_{x_2}^2, \cdots, V_{x_l}^l$. Similar to the argument in Section III-A, under such server placement, the latencies between all clients and their nearest servers are 1, and the latency between any two servers is also 1. Thus, the interaction path from a client to itself has length $1 + 1 = 2$, and the interaction path length between two distinct clients is bounded by $1 + 1 + 1 = 3$. Note that there are $n(n-1)$ pairs of distinct clients. Therefore, the total interaction path length is bounded by $3n(n-1) + 2n = 3n^2 - n = H$.

Next, we show that a valid server provisioning solution for instance $\mathcal{Y}$ gives rise to a set cover of size at most k

[3]The server locations not connected by any clients can simply be removed from S.

for instance $\mathcal{R}$. Let $S \subseteq V_S$ be a set of server locations that produces a total interaction path length not exceeding H. Without loss of generality, assume that the server at each location in S is connected by at least one client. Note that each node in V_S has latency 1 to exactly one client in V_C and has latency 2 to all the other clients. As a result, if two distinct clients in V_C connect to the same server in S, their interaction path length is at least $1 + 2 = 3$. On the other hand, the latency between any two nodes in V_S is at least 1. Thus, if two distinct clients connect to different servers in S, their interaction path length is at least $1 + 1 + 1 = 3$. Therefore, regardless of server placement, the interaction path length between any two distinct clients is at least 3. Since there are $n(n-1)$ pairs of distinct clients, this implies that the total interaction path length from all clients to themselves under server placement S cannot exceed $H - 3n(n-1) = 2n$.

Since the shortest latency from each client to the nodes in V_S is 1, it follows that the latencies from all the clients to their nearest servers must be exactly 1. Therefore, each element $p_i \in P$ is covered by the subset Q_j corresponding to the node cluster of c_i's nearest server. That is, $\mathbb{Q}' = \{Q_j \mid \exists i, V_j^i \cap S \neq \emptyset\}$ is a set cover for P.

To limit the total interaction path length by H, the interaction path length between any two distinct clients must now be exactly 3. This implies that the latency between any two servers in S must be 1. Thus, for each group of nodes V^i, all the servers $V^i \cap S$ must come from the same cluster of V^i. It follows that $|\mathbb{Q}'| \leq k$. Therefore, $\mathbb{Q}'$ is a set cover of size at most k.

In summary, there exists a set cover of size at most k for instance $\mathcal{R}$ if and only if a server placement with total interaction path length bounded by H can be found for instance $\mathcal{Y}$. Thus, the server provisioning problem with restricted choices of server locations is NP-hard.

C. Limited Number of Server Locations to Select

If the DIA operator can deploy only a limited number of servers due to budget restriction, a cap can be set on the number of server locations to select. If the cap is less than the number of clients, the server provisioning problem is also NP-hard. This can be proved by a polynomial reduction from the 3-Dimensional Matching (3DM) problem [10]. The decision version of the 3DM problem is defined as follows: Given three disjoint sets W, X and Y each having k elements, and a set of triples $M \subseteq W \times X \times Y$, find out whether M contains a 3-dimensional matching, i.e., whether there exists a subset $M' \subseteq M$ such that $|M'| = k$ and any two triples (w_a, x_a, y_a) and $(w_b, x_b, y_b) \in M'$ satisfy $w_a \neq w_b$, $x_a \neq x_b$ and $y_a \neq y_b$.

Given an instance $\mathcal{U}$ of the 3DM problem, we construct a network $G_{\mathcal{U}}$ consisting of the node set $V_C \cup V_M$. The set V_C contains $3k$ nodes, each corresponding to an element in $W \cup X \cup Y$ of instance $\mathcal{U}$. The set V_M contains the same

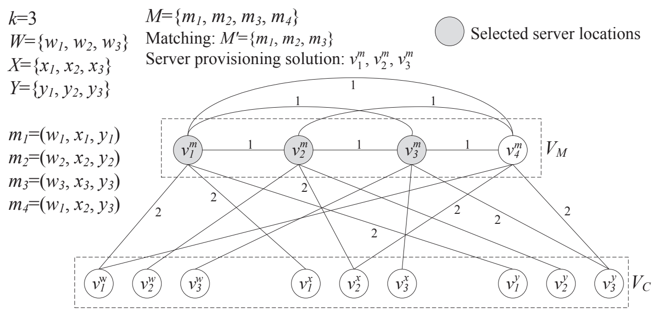

Figure 2. Example instances of the 3DM problem and the server provisioning problem

number of nodes as the size of M in instance $\mathcal{U}$. Each node in V_M corresponds to a triple in M, and establishes links of latency 2 to the three nodes in V_C corresponding to its three coordinates. For example, in Figure 2, node v_4^m, which corresponds to the triple $m_4 = (w_1, x_2, y_3)$, has three links to nodes v_1^w, v_2^x and v_3^y in V_C. In addition, all the nodes in V_M are inter-connected with each other via links of latency 1. Suppose that a client is located at each node in V_C, and all the nodes in network $G_{\mathcal{U}}$ are candidate server locations. An instance $\mathcal{F}$ of the server provisioning problem is then defined on network $G_{\mathcal{U}}$ by capping the number of server locations to select at k and setting a bound $A = 45k^2 - 9k$ on the total interaction path length between all client pairs.

We first prove that if M contains a matching for instance $\mathcal{U}$, there must exist a server placement with total interaction path length bounded by A for instance $\mathcal{F}$. Suppose that $M' \subseteq M$ is a matching. Let S be the set of k nodes in V_M that represent the triples in M' (for example, nodes v_1^m, v_2^m and v_3^m in Figure 2). We construct a server placement by selecting all the k nodes in S as the server locations. Then, each client in V_C has exactly one adjacent node in S. Thus, each client connects to the server placed at its adjacent node in S and the latency between them is 2. Since all the servers in S are inter-connected by links of latency 1, and each server has exactly three adjacent clients, the total interaction path length under placement S is given by

$$\begin{aligned}&(2+0+2)\cdot 9k + (2+1+2)\cdot(9k^2 - 9k)\\&= 45k^2 - 9k = A.\end{aligned}$$

Next, we prove that M contains a matching for instance $\mathcal{U}$ if a valid server placement solution can be found for instance $\mathcal{F}$. Consider a set S of up to k server locations in network $G_{\mathcal{U}}$. Let b_i ($i \geq 0$) be the number of clients that have latency i to their nearest servers in S. Since there are a total of $3k$ clients and all the links incident on clients have latency 2, we have $\sum_{i\geq 0} b_i = 3k$ and $b_1 = 0$. Note that there are b_0 servers of S located in V_C. So, the number of servers located in V_M is at most $k - b_0$. If a client has latency 2 to its nearest server, this server must be adjacent to the client and thus is located in V_M. Since each server in V_M has three adjacent clients, we have $b_2 \leq 3 \cdot (k - b_0)$. Thus, the total length of the interaction paths from the clients to themselves is

$$\begin{aligned}
\sum_{i\geq 0} 2i \cdot b_i &= 0 \cdot b_0 + 4 \cdot b_2 + \sum_{i\geq 3} 2i \cdot b_i \\
&\geq 4 \cdot b_2 + 6 \cdot \sum_{i\geq 3} b_i \qquad (1) \\
&= 4 \cdot b_2 + 6 \cdot (3k - b_2 - b_0) \\
&= 18k - 2 \cdot b_2 - 6 \cdot b_0 \\
&\geq 18k - 2 \cdot (3 \cdot (k - b_0)) - 6 \cdot b_0 \qquad (2) \\
&= 12k.
\end{aligned}$$

Since all the links incident on clients have latency 2 and the clients are not adjacent to each other, the shortest possible interaction path between two distinct clients has length 4. Let p_i $(i \geq 4)$ be the number of pairs of distinct clients whose interaction path length is i under server placement S. Since there are $3k(3k-1)$ pairs of distinct clients in total, we have $\sum_{i\geq 4} p_i = 3k(3k-1)$. If any two distinct clients have an interaction path length of 4, there must be a server in V_M that is adjacent to both clients. Each server in V_M has three adjacent clients and can thus support at most $3 \cdot 2 = 6$ interaction paths of length 4 between distinct clients. Since there are at most k servers, we have $p_4 \leq 6k$. Therefore, the total length of the interaction paths between distinct clients is

$$\begin{aligned}
\sum_{i\geq 4} i \cdot p_i &= 4 \cdot p_4 + \sum_{i\geq 5} i \cdot p_i \\
&\geq 4 \cdot p_4 + 5 \cdot \sum_{i\geq 5} p_i \qquad (3) \\
&= 4 \cdot p_4 + 5 \cdot (3k(3k-1) - p_4) \\
&= 45k^2 - 15k - p_4 \\
&\geq 45k^2 - 15k - 6k \qquad (4) \\
&= 45k^2 - 21k.
\end{aligned}$$

Thus, the total interaction path length is

$$\begin{aligned}
\sum_{i\geq 0} 2i \cdot b_i + \sum_{i\geq 4} i \cdot p_i &\geq 12k + 45k^2 - 21k \\
&= 45k^2 - 9k = A.
\end{aligned}$$

Therefore, if server placement S has a total interaction path length bounded by A, the equality must be satisfied at all the steps (1), (2), (3) and (4) in the above derivation.

Equality at step (3) implies that $\sum_{i\geq 5} p_i = 0$. So, at most one client can have latency 3 to its nearest server in S, i.e., $b_3 \leq 1$. Equality at step (1) indicates $\sum_{i\geq 3} b_i = 0$. Thus, $b_0 + b_2 + b_3 = 3k$. It follows from equality at step (2) (i.e., $b_2 = 3(k - b_0)$) that $b_3 = 2 \cdot b_0$, so b_3 is an even number. Therefore, b_3 must be 0, and b_0 is also 0. As a result, we have $b_2 = 3k$, i.e., the latency from each client to its nearest server in S must be 2. Thus, all the server locations in S are selected from node set V_M. Since each node in V_M has only three adjacent clients, exactly k server locations must be selected from V_M and these server locations cannot have any adjacent client in common. Therefore, the k triples of M corresponding to the k server locations selected in V_M form a matching.

Hence, M contains a matching for instance $\mathcal{U}$ if and only if there exists a server placement with total interaction path length bounded by A for instance $\mathcal{F}$. Thus, the server provisioning problem with limited number of server locations to select is NP-hard.

IV. GREEDY ALGORITHM

In this section, we present an efficient greedy algorithm called GREEDY for server provisioning. The computation of GREEDY is based simply on the network latencies between clients and candidate server locations, which can be acquired with existing tools like ping and King [19].

Without loss of generality, the GREEDY algorithm takes a parameter k that indicates the cap on the number of server locations to select. In the case where there is no limit on the number of server locations to select, k may simply be set to the number of candidate server locations. The GREEDY algorithm starts with an empty set of server locations S, and adds to S an unselected candidate location that leads to the maximum reduction in the total interaction path length in each iteration. The algorithm terminates when no new server location can be further selected to reduce the total interaction path length or the number of server locations selected reaches k.

Algorithm 1 shows the pseudo code of the GREEDY algorithm. In each iteration, the algorithm evaluates each unselected candidate location $s \in Z \setminus S$ and calculates the

```
1: $S \leftarrow \emptyset$, $D_{current} \leftarrow \infty$;
2: for all $c \in C$ do
3:     $n[c] \leftarrow \varnothing$; //nearest server to $c$
4: for $i = 1$ to $k$ do
5:     $D^* \leftarrow \infty$;
6:     for all $s \in Z \setminus S$ do
7:         for all $s_i \in S \cup \{s\}$ do
8:             $a[s_i] \leftarrow 0$; //number of clients connecting to $s_i$
9:         $D \leftarrow 0$; //total interaction path length when evaluating $s$
10:        for all $c \in C$ do
11:            if $n[c] = \varnothing$ or $d(c,s) < d(c,n[c])$ then
12:                $a[s] \leftarrow a[s] + 1;\ D \leftarrow D + 2 \cdot |C| \cdot d(c,s)$;
13:            else
14:                $a[n[c]] \leftarrow a[n[c]] + 1;\ D \leftarrow D + 2 \cdot |C| \cdot d(c,n[c])$;
15:        $D \leftarrow D + \sum_{s_i \in S \cup \{s\}} \sum_{s_j \in S \cup \{s\}} a[s_i] \cdot a[s_j] \cdot d(s_i, s_j)$;
16:        if $D < D^*$ then
17:            $D^* \leftarrow D$; $s^* \leftarrow s$;
18:    if $D^* \geq D_{current}$ then
19:        break; //stop if cannot further reduce interaction path length
20:    $S \leftarrow S \cup \{s^*\}$; $D_{current} \leftarrow D^*$;
21:    for all $c \in C$ do
22:        if $n[c] = \varnothing$ or $d(c,s^*) < d(c,n[c])$ then
23:            $n[c] \leftarrow s^*$; //update the nearest server of client $c$
24: output $S$;
```

Algorithm 1: GREEDY Server Provisioning Algorithm

total interaction path length D if s is added to S (lines 6 to 17). The candidate location that results in the minimum value of D and the corresponding D value are recorded in s^* and D^* respectively (lines 16 to 17). At the end of each iteration, the algorithm keeps the nearest servers of all clients under the current placement S in an array $n[\cdot]$ (lines 21 to 23). When evaluating a candidate server location s in the next iteration, we can find out the nearest server of each client c by simply comparing the latency from c to $n[c]$ with the latency from c to s (lines 11 to 14). In this way, the efficiency of the algorithm is improved by avoiding repeatedly comparing the latencies between each client and all the selected server locations.

Suppose that under a server placement S, the number of clients connecting to each server $s_i \in S$ is $m(s_i)$. Then, the total interaction path length between all client pairs can be rewritten as

$$\begin{aligned}
&\sum_{c\in C}\sum_{c'\in C}\Big(d(c,n(c,S)) + d(n(c,S),n(c',S)) \\
&\qquad + d(n(c',S),c')\Big) \\
&= |C|\cdot\sum_{c\in C} d(c,n(c,S)) + |C|\cdot\sum_{c'\in C} d(n(c',S),c') \\
&\quad + \sum_{c\in C}\sum_{c'\in C} d(n(c,S),n(c',S)) \\
&= 2|C|\sum_{c\in C} d(c,n(c,S)) \\
&\quad + \sum_{s_i\in S}\sum_{s_j\in S} m(s_i)\cdot m(s_j)\cdot d(s_i,s_j). \qquad (5)
\end{aligned}$$

Thus, to calculate the total interaction path length resulting from adding a candidate location s to S, the GREEDY algorithm first counts the number of clients connecting to each server $s_i \in S\cup\{s\}$ and records it in $a[s_i]$. Then, the total interaction path length is calculated according to the above formula (5) (lines 10 to 15). The computational complexity of the total interaction path length is $O(|C|+k^2)$. Since the number of candidate server locations to evaluate in each iteration is capped by $|Z|$ (where Z is the set of candidate server locations) and there are at most k iterations, the total time complexity of the GREEDY algorithm is $O(k|Z|(|C|+k^2))$.

When assuming that the network latency satisfies the triangle inequality, we can show that the above GREEDY algorithm has an approximation ratio of 2, irrespective of whether the choices of server locations are restricted and the cap on the number of server locations to select (if any).

Theorem 2. *For networks with the triangle inequality, the* GREEDY *algorithm produces a total interaction path length within two times of that in an optimal server placement.*

Proof: Suppose that an optimal server placement selects h server locations s_1, s_2, $\cdots$, and s_h. For each location s_i, denote by C_i the set of clients connecting to the server at s_i under the optimal placement. It is obvious that $\bigcup_{i=1}^{h} C_i = C$, where C is the set of all clients. According to (5), the total interaction path length under the optimal server placement is given by

$$OPT = 2|C|\sum_{j=1}^{h}\sum_{c\in C_j} d(c,s_j) + \sum_{i=1}^{h}\sum_{j\neq i}|C_i||C_j|d(s_i,s_j).$$

If only one server is placed at a location s in the network, all the clients would connect to that server and thus the total interaction path length is given by

$$\sum_{c\in C}\sum_{c'\in C}(d(c,s)+d(s,s)+d(s,c')) = 2|C|\cdot\sum_{c\in C} d(c,s).$$

Suppose that s^* is the first server location selected by the GREEDY algorithm. Since the GREEDY algorithm always selects the best known server location in each iteration, the total interaction path length resulting from selecting s^* cannot be longer than those resulting from selecting s_1, s_2, $\cdots$, or s_h in the first iteration, i.e.,

$$2|C|\cdot\sum_{c\in C} d(c,s^*) \le 2|C|\cdot\sum_{c\in C} d(c,s_1),$$

$$2|C|\cdot\sum_{c\in C} d(c,s^*) \le 2|C|\cdot\sum_{c\in C} d(c,s_2),$$

$$\cdots\cdots$$

and

$$2|C|\cdot\sum_{c\in C} d(c,s^*) \le 2|C|\cdot\sum_{c\in C} d(c,s_h).$$

Taking a weighted average of the interaction path lengths on the right sides of the above inequalities, we have

$$\begin{aligned}
&2|C|\cdot\sum_{c\in C} d(c,s^*) \le \frac{1}{|C|}\cdot\sum_{i=1}^{h}\Big(|C_i|\cdot 2|C|\cdot\sum_{c\in C} d(c,s_i)\Big) \\
&= 2\cdot\sum_{i=1}^{h}\Big(|C_i|\cdot\sum_{c\in C} d(c,s_i)\Big) \\
&= 2\cdot\sum_{i=1}^{h}\Big(|C_i|\cdot\Big(\sum_{c\in C_i} d(c,s_i) + \sum_{j\neq i}\sum_{c\in C_j} d(c,s_i)\Big)\Big).
\end{aligned}$$

By the triangle inequality, for each client $c \in C_j$, we have

$$d(c,s_i) \le d(c,s_j) + d(s_j,s_i).$$

Therefore,

$$\begin{aligned}
&2|C|\cdot\sum_{c\in C} d(c,s^*) \\
&\le 2\cdot\sum_{i=1}^{h}\Big(|C_i|\cdot\Big(\sum_{c\in C_i} d(c,s_i) \\
&\quad + \sum_{j\neq i}\sum_{c\in C_j}\big(d(c,s_j)+d(s_j,s_i)\big)\Big)\Big)
\end{aligned}$$

$$= 2 \cdot \sum_{i=1}^{h} \Bigg(|C_i| \Big(\sum_{j=1}^{h} \sum_{c \in C_j} d(c, s_j) + \sum_{j \neq i} |C_j| d(s_j, s_i) \Big) \Bigg)$$
$$= 2 \sum_{i=1}^{h} |C_i| \sum_{j=1}^{h} \sum_{c \in C_j} d(c, s_j) + 2 \sum_{i=1}^{h} \sum_{j \neq i} |C_i||C_j| d(s_j, s_i)$$
$$= 2|C| \sum_{j=1}^{h} \sum_{c \in C_j} d(c, s_j) + 2 \sum_{i=1}^{h} \sum_{j \neq i} |C_i||C_j| d(s_i, s_j)$$
$$\leq 2 \Bigg(2|C| \sum_{j=1}^{h} \sum_{c \in C_j} d(c, s_j) + \sum_{i=1}^{h} \sum_{j \neq i} |C_i||C_j| d(s_i, s_j) \Bigg)$$
$$= 2 \cdot OPT.$$

This implies that the total interaction path length on selecting the first server location in the GREEDY algorithm is within two times of that in an optimal server placement. Since the GREEDY algorithm adds a new server location in each subsequent iteration only if it reduces the total interaction path length, the total interaction path length eventually produced by GREEDY must also be within two times of the optimum. Hence, the theorem is proven. ■

The approximation ratio 2 of the GREEDY algorithm is tight. Figure 3 presents a tight example in which the network contains two client groups $C_A = \{c_1, c_2, c_3\}$ and $C_B = \{c_4, c_5, c_6\}$, and an additional node g. The latency between any two nodes in the same client group is 1. The latency between any two nodes in different client groups is $\frac{4}{3} + 3\delta$, where $\delta > 0$. All the nodes in C_A and C_B have latencies $1+\delta$ to node g. It is easy to infer that the network satisfies the triangle inequality when $\delta \leq \frac{2}{3}$. Suppose that all the nodes in the network are candidate server locations and there is no limit on the number of server locations to select. Then, the optimal placement is to select all the nodes in $C_A \cup C_B$ as server locations. Under this placement, each client connects to the server co-located with it, so the total interaction path length is $OPT = \sum_{i=1}^{6} \sum_{j \neq i} d(c_i, c_j) = 12 \cdot 1 + 18 \cdot (\frac{4}{3} + 3\delta) = 36 + 54\delta$. On the other hand, the first server location selected by the GREEDY algorithm is g. This is because placing a server at g leads to the total interaction path length of $D_1 = 12 \cdot \sum_{i=1}^{6} d(c_i, g) = 72 + 72\delta$, whereas if a server is

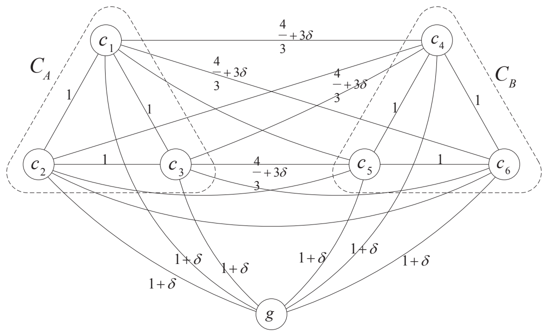

Figure 3. The approximation ratio 2 of the GREEDY algorithm is tight.

placed at a node in $C_A \cup C_B$, the total interaction path length is $12 \cdot (2 \cdot 1 + 3 \cdot (\frac{4}{3} + 3\delta)) = 72 + 108\delta > D_1$. In the second iteration of the GREEDY algorithm, if a new server location c is further selected from $C_A \cup C_B$, all the three clients in c's group would connect to the server at c and the other three clients remain connecting to the server at g. Therefore, the total interaction path length becomes $D_2 = 12 \cdot (2 \cdot 1 + 3 \cdot (1 + \delta)) + 18 \cdot (1 + \delta) = 78 + 54\delta$. When $\delta < \frac{1}{3}$, we have $D_2 > D_1$. Thus, the GREEDY algorithm does not add any new server location and terminates at the second iteration. As a result, the final set of server locations selected is $\{g\}$. When δ approaches 0, the ratio between the interaction path lengths produced by the GREEDY algorithm and the optimal server placement is $\lim_{\delta \to 0} \frac{D_1}{OPT} = \lim_{\delta \to 0} \frac{72+72\delta}{36+54\delta} = 2$.

V. EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of the proposed GREEDY algorithm by using the Meridian dataset [20], which contains the pairwise latency measurements between 2500 nodes in the real Internet. The measurements for some node pairs are not available in the dataset. We discard the nodes involved in the missing measurements, and use the complete pairwise latency matrix among the remaining 1796 nodes as our simulated network. A client is assumed to be located at each node in the network.

To the best of our knowledge, there is hardly any server provisioning algorithm tuned for DIAs. For the purpose of comparison, we also implement a k-median heuristic [7] and a k-center heuristic [8] in our experiments. Both heuristics work for k iterations. In each iteration, the k-median heuristic adds a new server location that results in the largest reduction in the total latency from the clients to their nearest servers. Similarly, the k-center heuristic adds a new server location that results in the lowest value of the maximum latency from the clients to their nearest servers.

To quantify the relative performance of the algorithms, we normalize the total interaction path lengths produced by different algorithms by a theoretical lower bound, which is derived as follows. Given a set Z of candidate server locations, the shortest possible interaction path length between two clients c_i and c_j is given by $\min_{s_a, s_b \in Z}(d(c_i, s_a) + d(s_a, s_b) + d(s_b, c_j))$. Therefore, the total length of interaction paths between all client pairs has a lower bound of

$$\sum_{c_i \in C} \sum_{c_j \in C} \Big(\min_{s_a, s_b \in Z} \big(d(c_i, s_a) + d(s_a, s_b) + d(s_b, c_j) \big) \Big). \quad (6)$$

This bound is a super-optimum that may not be achievable by any real server placement, because it does not enforce each client to connect to its nearest server or even a single server through which it interacts with all the other clients. The total interaction path length normalized by the above bound shall be called the *normalized interactivity*.

In the first experiment, we assume that all 1796 nodes in the network are candidate server locations. Figure 4 shows

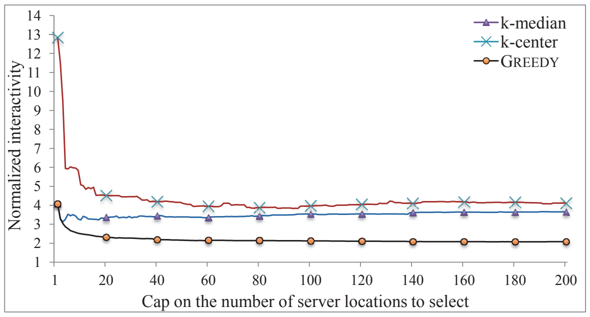

Figure 4. Normalized interactivity of different algorithms when all nodes are candidate server locations

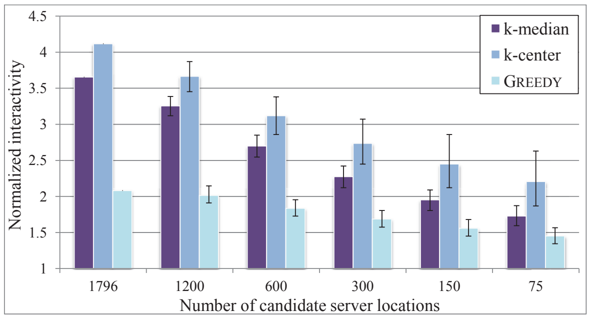

Figure 5. Normalized interactivity of different algorithms for different numbers of candidate server locations

the performance of the three algorithms for different caps on the number of server locations to select. Since each client connects to its nearest server, the latencies from the clients to their connected servers normally decrease with increasing number of server locations selected. Thus, the k-median and k-center heuristics always place servers up to the cap number. However, aggressively reducing client-to-server latencies alone may result in longer latencies between the servers of different clients and consequently increase the interaction path length between clients. As seen from Figure 4, the interaction path lengths of k-median and k-center placements do not always reduce with increasing number of server locations selected because they do not consider inter-server latencies in server provisioning. In contrast, our GREEDY algorithm stops selecting new server locations when the total interaction path length cannot be further reduced. Therefore, its interactivity performance does not deteriorate when a larger number of server locations are allowed to be selected. In our simulated network, the GREEDY algorithm does not select additional server locations beyond 195. Figure 4 shows that the GREEDY algorithm significantly outperforms the k-median and k-center placements.[4] This implies that cutting down the client-to-server latency alone is not very effective for improving the interactivity of DIAs.

In the second experiment, we restrict the choices of server locations in the network. We randomly choose subsets of 1200, 600, 300, 150 and 75 nodes from the network and execute the algorithms using these subsets as the candidate server locations. In this experiment, no limit is set on the number of server locations for our GREEDY algorithm to select. We run the GREEDY algorithm until it terminates and record the number of server locations actually selected by GREEDY. This number is then used as the number of server locations to select in executing k-median and k-center placements. In this way, all the three algorithms select the same number of server locations in the network to allow for fair comparison. For each set size, we perform 1000 simulation runs using 1000 different sets of candidate server locations chosen at random. Figure 5 shows the average performance of the algorithms together with the 90th and 10th percentile results for different numbers of candidate server locations. Here, the total interaction path lengths are normalized by the lower bound (6) derived using the respective sets of candidate server locations chosen. Thus, the results of various simulation runs are normalized by different lower bound values. For comparison purpose, we also include the results when there is no restriction on the choices of server locations.[5] It can be seen from Figure 5 that our GREEDY algorithm consistently results in much better interactivity than k-median and k-center placements. It can also be observed that the normalized interactivity of the algorithms generally improves when the number of candidate server locations reduces. This is because fewer candidate server locations imply smaller search space, so that it is more likely to hit a good solution within certain number of attempts.

We also normalize the total interaction path lengths of different algorithms by the lower bound (6) derived by setting Z as all 1796 nodes in the network. In this way, the results of all simulation runs are normalized by the same lower bound value. Figure 6 shows the results with such normalization. It can be seen that when the number of candidate server locations reduces, the new normalized interactivity increases for all the three algorithms. This is intuitive since a smaller candidate set restricts the choices of server locations more severely, thereby degrading the absolute interactivity performance.

Table I reports the average running times of the algorithms for different numbers of candidate server locations, which are measured on a machine with Intel Xeon Quad Core 2.93GHz CPU and 4GB RAM. In general, the running time of our GREEDY algorithm is on the same order of magnitude as the k-median and k-center placements. It is worth noting that server provisioning is often planned on mid- to long-term basis since deploying new servers

[4]The approximation ratio 2 of GREEDY does not apply to our simulations because the lower bound we calculate is a super-optimum and real Internet latency data do not necessarily satisfy the triangle inequality.

[5]No percentile result is plotted in this case because the simulation is run just once with all 1796 nodes in the network as candidate server locations.

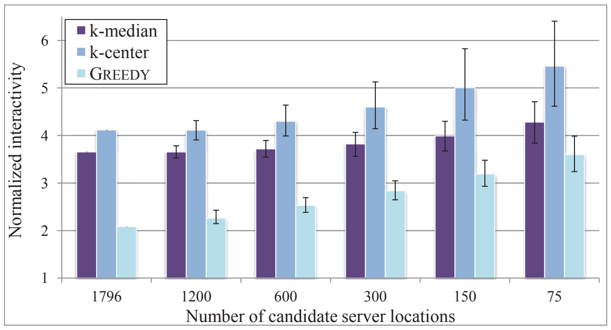

Figure 6. Performance of different algorithms normalized by the same lower bound value

Table I
AVERAGE RUNNING TIMES OF DIFFERENT ALGORITHMS (IN SECONDS)

candidate server locations	k-median	k-center	GREEDY
1796	16.926s	16.911s	44.991s
1200	10.551s	10.379s	17.406s
600	3.520s	3.494s	3.510s
300	0.846s	0.851s	0.944s
150	0.294s	0.320s	0.297s
75	0.144s	0.122s	0.117s

may involve amendment to hardware infrastructure or lease agreements with third-party service providers. Therefore, the running time of our GREEDY algorithm is very acceptable.

VI. CONCLUSION

In this paper, we have investigated the server provisioning problem for DIAs with the objective of reducing the network latency involved in the interaction between clients. We have proven that this problem is NP-hard when (a) the network latency does not satisfy the triangle inequality; or (b) the choices of server locations in the network are restricted; or (c) the number of server locations to select is limited. We have presented a GREEDY heuristic algorithm for server provisioning and conducted a theoretical analysis of its approximation ratio. Experimental evaluations are also conducted using real Internet latency data. The results show that aggressively reducing the client-to-server latency alone is not effective for improving the interactivity of DIAs. Our proposed GREEDY algorithm substantially outperforms traditional k-median and k-center server placements.

ACKNOWLEDGEMENT

This research is supported by Multi-plAtform Game Innovation Centre (MAGIC) in Nanyang Technological University. MAGIC is funded by the Interactive Digital Media Programme Office (IDMPO) hosted by the Media Development Authority of Singapore.

REFERENCES

[1] M. Claypool and K. Claypool. Latency Can Kill: Precision and Deadline in Online Games. In *Proc. ACM MMSys*, pages 215–222, 2010.

[2] R. B. Jennings, E. M. Nahum, D. P. Olshefski, D. Saha, Z.-Y. Shae and C. Waters. A Study of Internet Instant Messaging and Chat Protocols. *IEEE Network*, 20(4):16–21, 2006.

[3] Agustina, F. Liu, S. Xia, H. Shen and C. Sun. CoMaya: Incorporating Advanced Collaboration Capabilities into 3D Digital Media Design Tools. In *Proc. ACM CSCW*, pages 5–8, 2008.

[4] L. Ahmad, A. Boukerche, A. Al Hamidi, A. Shadid and R. Pazzi. Web-Based e-Learning in 3D Large Scale Distributed Interactive Simulations using HLA/RTI. In *Proc. IEEE IPDPS*, pages 1–4, 2008.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia. A View of Cloud Computing. *Communications of the ACM*, 53(4):50–58, 2010.

[6] F. Safaei, P. Boustead, C. D. Nguyen, J. Brun and M. Dowlatshahi. Latency-Driven Distribution: Infrastructure Needs of Participatory Entertainment Applications. *IEEE Communications Magazine*, 43(5):106–112, 2005.

[7] L. Qiu, V. N. Padmanabhan and G. M. Voelker. On the Placement of Web Server Replicas. In *Proc. IEEE INFOCOM*, pages 1587–1596, 2001.

[8] E. Cronin, S. Jamin, J. Cheng, A. R. Kurc, D. Raz and Y. Shavitt. Constrained Mirror Placement on the Internet. *IEEE Journal on Selected Areas in Communications*, 20(7):1369–1382, 2002.

[9] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis and A. Bestavros. Distributed Placement of Service Facilities in Large-Scale Networks. In *Proc. IEEE INFOCOM*, pages 2144–2152, 2007.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[11] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala and V. Pandit. Local Search Heuristic for k-Median and Facility Location Problems. In *Proc. ACM STOC*, pages 21–29, 2001.

[12] M. Chrobak, C. Kenyon and N. Young. The Reverse Greedy Algorithm for the Metric k-Median Problem. *Information Processing Letters*, 97(2):68–72, 2006.

[13] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[14] L. Zhang and X. Tang. Optimizing Client Assignment for Enhancing Interactivity in Distributed Interactive Applications. *IEEE/ACM Transactions on Networking*, 20(6):1707–1720, 2012.

[15] K.-W. Lee, B.-J. Ko and S. Calo. Adaptive Server Selection for Large Scale Interactive Online Games. *Computer Networks*, 49(1):84–102, 2005.

[16] S. D. Webb, S. Soh and W. Lau. Enhanced Mirrored Servers for Network Games. In *Proc. ACM NetGames*, pages 117–122, 2007.

[17] C. Ding, Y. Chen, T. Xu and X. Fu. CloudGPS: A Scalable and ISP-Friendly Server Selection Scheme in Cloud Computing Environments. In *Proc. IEEE/ACM IWQoS*, 2012.

[18] C. Lumezanu, R. Baden, N. Spring and B. Bhattacharjee. Triangle Inequality and Routing Policy Violations in the Internet. In *Proc. PAM*, pages 45–54, 2009.

[19] K. P. Gummadi, S. Saroiu and S. D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proc. ACM SIGCOMM Workshop on Internet Measurement*, pages 5–18, 2002.

[20] The Meridian Latency Data Set. http://www.cs.cornell.edu/People/egs/meridian/.