# The Server Provisioning Problem for Continuous Distributed Interactive Applications

Hanying Zheng and Xueyan Tang

***Abstract*—In this paper, we study the server provisioning problem for continuous Distributed Interactive Applications (DIAs) whose application states not only change because of the operations performed by participants, but also evolve along with the passing of time. We focus on finding the locations of servers for hosting DIAs, with the goals of optimizing the interactivity performance while fulfilling the consistency and fairness requirements. We show that the server provisioning problem is challenging by presenting its NP-hardness and non-approximability results under several conditions. We propose two efficient server placement algorithms and analyze their approximation ratios. The approximation ratio of the proposed M-BETTER algorithm is quite close to a lower bound for any polynomial-time algorithm. We also conduct experimental evaluations to compare the proposed algorithms with several baseline server placements.**

***Index Terms*—Distributed interactive application, server placement, interactivity, approximation algorithm**

## I. INTRODUCTION

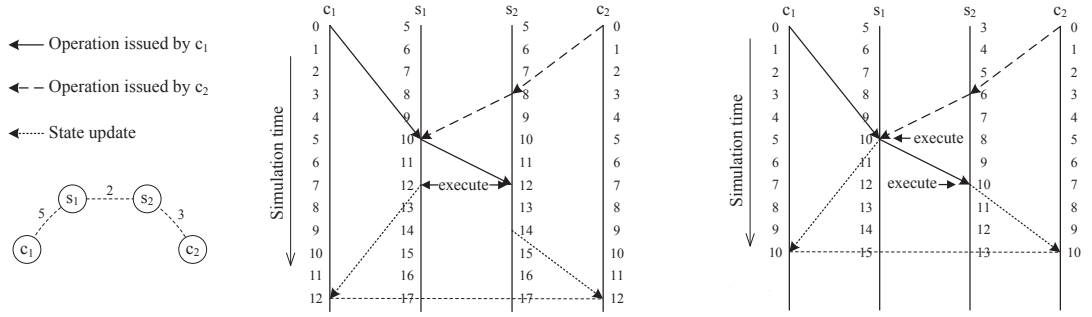
Distributed Interactive Applications (DIAs) are emerging technologies that open up new opportunities for geographically distributed participants to interact with each other via computer networks. Examples of DIAs include networked gaming [1], [2], distributed e-learning [3], and collaborative computer-aided design and engineering [4]. DIAs usually operate with client-server architectures [5] in which the *servers* maintain the application states, and execute the operations submitted by the participants who are known as *clients*. Owing to different ways of maintaining the application states, DIAs can be classified into two categories: discrete DIAs and continuous DIAs [6]. In discrete DIAs, the application states are only updated due to the operations performed by clients. In continuous DIAs, the application states not only change because of executing the client-initiated operations, but also evolve along with the elapse of time. A typical example of continuous DIAs is networked gaming, where the states of virtual game worlds are often updated at a fast pace even when there is no operation input from the players.

A major barrier to the quality of experience in DIAs is the communication latency across the network. Wide geographical spreads of clients in large-scale DIAs necessitate distributed deployment of servers to support the interactions among clients [5]. Servers can be placed not only at locations in the center of the network (e.g. cloud data centers), but also near the edges of the network such as nano data centers [7] and smart edge nodes [8]. The increasing elasticity of computing resources at these locations allows the DIA operators to quickly scale up or down the capacity on demand for hosting their applications. However, even with distributed server infrastructures, the network latency cannot be completely eliminated from the interactions between clients in DIAs. The network latency involved in client interactions is directly affected by the locations where servers are placed. Thus, server placement is of crucial importance to the interactivity performance of DIAs.

Besides interactivity, the network latency also challenges the consistency and fairness of DIAs. Consistency means to create shared common views of the application state among all clients to enable meaningful interactions [9]. Significant divergence of the application state can seriously affect the behavior and decisions of the participants. Fairness is to ensure that all clients have equal chances to participate despite their geographically dispersed locations in the network [10], [11]. This is particularly important for those DIAs in which the participants compete with each other intensely, such as many online games. Due to constant state updates along with time passing, fulfilling the consistency and fairness requirements is much tougher in continuous DIAs than in discrete DIAs. As shall be elaborated later, the higher time-sensitivity of continuous DIAs entails additional synchronization delays in the interactions among clients for maintaining consistency and fairness. In our recent work [12], [13], we have studied server provisioning for discrete DIAs in which no synchronization delay is needed. In contrast, this paper explores server provisioning for continuous DIAs with consideration of synchronization delays, which lead to a different optimization objective from that for discrete DIAs. From the computability perspective, the difference in the optimization objective gives rise to a much richer set of non-approximability results in this paper than in [12], [13]. New approaches are also required to design algorithms and analyze their approximability for continuous DIAs.

In this paper, we formally define the problem of finding the locations of servers for hosting continuous DIAs, with the goal of optimizing the interactivity performance while maintaining the consistency and fairness of DIAs. Inspired by the analysis in [12], [13], we show the NP-hardness of the problem under *any* one of the following conditions which may be prevalent in practice: (i) the network latencies do not satisfy the triangle inequality; or (ii) the locations where servers can be placed are restricted; or (iii) the number of server locations to select is limited. We further prove that the server provisioning problem cannot be approximated within any bounded factor under condition (i), within a factor of $3/2$ under condition (ii), and within a factor of $4/3$ under condition

The authors are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798.




(a) Interaction time when all servers have synchronized simulation times (b) Minimum achievable interaction time

Fig. 1. An example of minimum achievable interaction time.

(iii). We propose two efficient server placement algorithms and show that they significantly outperform the baseline server placements by means of both theoretical analysis and experimental evaluation. In particular, the proposed M-BETTER algorithm has an approximation ratio quite close to the lower bound of $3/2$.

The rest of this paper is organized as follows. Section II formulates the server provisioning problem for continuous DIAs. Its hardness and non-approximability results are presented in Section III. Section IV studies the approximability of the classical $k$-center server placement. Sections V and VI propose two server provisioning algorithms and analyze their approximation ratios. We experimentally evaluate the proposed algorithms using real Internet latency data in Section VII. Section VIII summarizes the related work. Finally, Section IX concludes the paper.

## II. PROBLEM FORMULATION

We model the underlying network of the DIA as a graph, whose node set includes a set of clients $C$ and a set of servers $S$. A distance $d(u, v)$ is associated with each pair of nodes $u$ and $v$, representing the network latency of the routing path between them.

Each client is connected to a server to participate in the DIA and interact with other clients [14], [15]. When a client performs an operation, the application state of the DIA should be updated accordingly by executing the operation at each server and then the state update should be disseminated to all clients to reflect the change. Consider the interaction between two clients $c$ and $c'$. In order to notify $c'$ of the change in the application state caused by an operation initiated by $c$, $c$ should first send its operation to its connected server $s$. Then, $s$ would forward the operation to the connected server $s'$ of client $c'$ if $s$ and $s'$ are different. After receiving the operation, $s'$ executes it and sends the resultant state update to $c'$ to complete the interaction. Therefore, the interaction involves the paths from $c$ to $s$, from $s$ to $s'$, and from $s'$ to $c'$. We define the concatenation of these three paths as the *interaction path* from $c$ to $c'$. The network latency involved in the interaction is given by the length of the interaction path, i.e., $d(c, s) + d(s, s') + d(s', c')$. Note that the interaction path from a client $c$ to itself has length $2d(c, s)$, which is the network latency involved for $c$ to see the effect of its own operation.

In continuous DIAs, the application states change due to not only client-initiated operations but also time passing [6]. Thus, the progress of the application state is often measured along a synthetic timescale known as the *simulation time* (for example, the time elapsed in the virtual game world of an online game) [16]. To ensure that all clients consistently observe the same effect caused by a user operation, the operation must be executed by all servers at the same simulation time. As a result, the servers may not be able to execute user-initiated operations immediately upon receiving them. They may have to wait and give enough time for the operations to be delivered to other servers [5], [15], [17]. In addition, fairness is concerned with the order of executing user-initiated operations [11]. To guarantee that all clients have equal chance of participation regardless of their network conditions, the user operations must be executed in the simulation time order of their initiations at the clients. The execution of operations must also preserve the simulation time interval between the operation initiations. Due to diverse network latencies, maintaining consistency and fairness in continuous DIAs introduce artificial synchronization delays in the interactions among clients.

Zhang et al. [16] have proved that the minimum achievable interaction time between clients for fulfilling the consistency and fairness requirements is given by the length of the longest interaction path among all clients. Figure 1 shows a simple example to illustrate the minimum achievable interaction time. Suppose that two clients $c_1$ and $c_2$ are connected to servers $s_1$ and $s_2$ respectively. The latencies from $c_1$ to $s_1$, from $c_2$ to $s_2$, and the inter-server latency are 5, 3 and 2 respectively. A straightforward simulation time setting is to synchronize the simulation times at the two servers (see Figure 1(a)). In this case, if an operation is issued at simulation time 0, considering the two possible initiating clients, the latest possible time for all the servers to receive the operation is at simulation time 12. Thus, the earliest simulation time for the servers to execute the operation is 12 in order to guarantee the consistency and fairness. Therefore, the resultant state update can only be seen by the clients after a time lag of 12. On the other hand, if the relative offset of simulation times at $s_1$ and $s_2$ is set to 2 as shown in Figure 1(b), all the servers would be able to receive an operation issued at simulation time 0 by time 10 at the latest, irrespective of the client initiating the operation.

As a result, the interaction time is reduced to 10. This is the minimum achievable interaction time, which is equal to the maximum interaction path length.

In many DIAs, the clients are naturally connected to their nearest servers, i.e., the servers having the shortest network latency to them [2], [14], [15], [18]. Thus, given a set of clients, the longest interaction path is primarily determined by where the servers are placed to support client interactions. Suppose that there is a set of candidate server locations $Z$ in the network where servers can be placed. The server provisioning problem for continuous DIAs is to find a set of locations $S \subseteq Z$ to place servers so that the maximum interaction path length among all clients, i.e.,

$$\max_{c,c' \in C} \{d(c, n(c,S)) + d(n(c,S), n(c',S)) + d(n(c',S), c')\},$$

is minimized,[1] where $n(c,S)$ denotes client $c$'s nearest server in $S$. To focus on reducing the network latency involved in the interaction, this basic problem definition does not assume any server capacity limitation. Extending our proposed algorithms to handle server capacity restrictions shall be discussed in Section VII-B.

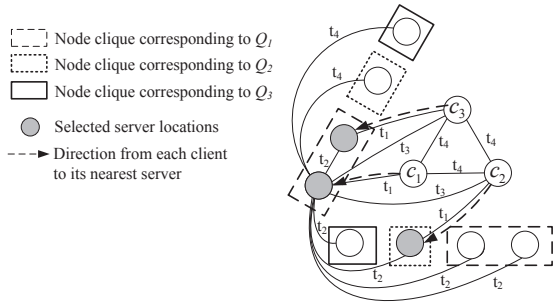
## III. Hardness of the Problem

If the network latency satisfies the triangle inequality, for any pair of clients $c$ and $c'$, it is obvious that $d(c,c') \leq d(c,s) + d(s,s') + d(s',c')$ for any two servers $s$ and $s'$ regardless of their locations. Thus, the best arrangement in this case is to have a server co-located with each client so that all the clients connect to their co-located servers, producing the shortest possible interaction path length between each pair of clients. However, Internet routing is not optimal in terms of network latency due to business-oriented strategies [19]. Normally, it is also impossible to place servers at the same locations as clients. For example, DIA operators who cannot afford to set up their own server infrastructures may have to run servers at the data centers operated by cloud providers. In addition, there might also be a limit on the number of server locations that a DIA operator can choose due to its budget constraints. Following similar methodology to that in [12], [13], [16], we show that under *any* one of the above conditions, the server provisioning problem for continuous DIAs is NP-hard.

### A. Hardness for Networks without Triangle Inequality

In the case that the network latency does not satisfy the triangle inequality, the NP-hardness of the server provisioning problem for continuous DIAs can be proved by a polynomial reduction from the minimum set cover problem which is NP-hard [20]. The decision version of the set cover problem is defined as follows: Given a finite set $P = \{p_1, p_2, \cdots, p_n\}$ and a collection $\mathbb{Q} = \{Q_1, Q_2, \cdots, Q_m\}$ of its subsets, and a positive integer $k \leq m$, decide whether $\mathbb{Q}$ contains a subcollection $\mathbb{Q}'$ of at most $k$ subsets such that $\bigcup_{Q \in \mathbb{Q}'} Q = P$.

[1]This optimization objective is different from that for discrete DIAs. Without synchronization delays, the interactivity performance of discrete DIAs is measured by the average interaction path length between all client pairs [12], [13].




$P = \{p_1, p_2, p_3\}$, $Q_1 = \{p_1, p_3\}$, $Q_2 = \{p_2\}$, $Q_3 = \{p_3\}$
$\mathbb{Q} = \{Q_1, Q_2, Q_3\}$, $k = 2$. Set Cover: $\mathbb{Q}' = \{Q_1, Q_2\}$

Fig. 2. Example instances of the set cover problem and the server provisioning problem for continuous DIAs

Given an instance $\mathcal{R}$ of the set cover problem, we first construct a network comprising a set $V_C$ of $n$ nodes and a set $V_S$ of $k$ disjoint groups of nodes as shown in Figure 2. The $n$ nodes of $V_C$ map onto to the elements in the set $P$ of instance $\mathcal{R}$. Each node group of $V_S$ contains $m$ cliques of sizes $|Q_1|, |Q_2|, \ldots,$ and $|Q_m|$, each of which corresponds to a subset in the collection $\mathbb{Q}$ by mapping each node in the clique onto an element in the corresponding subset.

In the constructed network, any two nodes in different groups of $V_S$ have a latency $t_2$ to each other. Any two nodes in the same group of $V_S$ have a latency $t_2$ if they belong to the same clique or map onto the same element in $P$. Otherwise, they have a latency $t_4$. For any two nodes $u \in V_S$ and $v \in V_C$, they have a latency $t_1$ if both $u$ and $v$ map onto the same element in $P$. Otherwise, they have a latency $t_3$. Any two nodes in $V_C$ have a latency $t_4$ to each other. For ease of reading, Figure 2 illustrates only the network latencies relevant to a representative node in $V_S$ as well as the latencies among the nodes in $V_C$.

We assume that

- $t_4 > 2t_1 + t_2$,
- $t_3 > t_1 + t_2$.

The second inequality implies that the constructed network does not satisfy the triangle inequality due to a triangle with sides $t_1$, $t_2$ and $t_3$ as shown in Figure 2.

We then define an instance $\mathcal{T}$ of the server provisioning problem in its decision version as follows: Given the constructed network, assuming that a client is located at each node of $V_C$ and all the nodes in the network (i.e., $V_C \cup V_S$) are candidate server locations, can we select a set of server locations to bound the maximum interaction path length among all clients by $2t_1 + t_2$?

We first prove that if there is an acceptable set cover for instance $\mathcal{R}$, then there must exist a valid solution for the server provisioning instance $\mathcal{T}$. Suppose that $\mathbb{Q}' = \{Q_{x_1}, Q_{x_2}, \cdots, Q_{x_l}\}$ (where $1 \leq l \leq k$) is a set cover. We can construct a server placement $S$ by selecting all the nodes in the $x_1$-th clique in the first group of $V_S$, the $x_2$-th clique in the second group of $V_S$, ..., and the $x_l$-th clique in the $l$-th group of $V_S$ as server locations (see Figure 2 for an example). Note that each element of $P$ is covered by at least one subset in $\mathbb{Q}'$. Therefore, for each client in $V_C$, at least one clique selected

by $S$ should contain a server having latency $t_1$ to the client. Our assumption $t_3 > t_1 + t_2$ implies that $t_3 > t_1$. So, each client has latency $t_1$ to its nearest server in $S$. Furthermore, placement $S$ selects at most one clique from each group of $V_S$. Thus, all the inter-server latencies in placement $S$ are $t_2$. As a result, the maximum interaction path length between any two clients is $2t_1+t_2$. Therefore, server placement $S$ is a valid solution for instance $\mathcal{T}$.

Next, we prove that if there is a valid solution for the server provisioning instance $\mathcal{T}$, there must exist a set cover of size at most $k$ for instance $\mathcal{R}$. We start by showing that in any valid solution of instance $\mathcal{T}$, each client and its nearest server must map onto the same element in $P$. If a client and its nearest server do not map onto the same element, the latency between them would be $t_3$ if $s \in V_S$ or would be $t_4$ if $s \in V_C$. Then, the interaction path from the client to itself is $2t_3$ or $2t_4$ long, which exceeds the bound $2t_1 + t_2$ since $t_3 > t_1 + t_2$ and $t_4 > 2t_1 + t_2$. Therefore, each client and its nearest server must map onto the same element. This implies that: (a) if the nearest server $s$ of a client $c$ is located in $V_C$, the latency between $c$ and $s$ can only be 0, i.e., $s$ and $c$ must be located at the same node; (b) if the nearest server $s$ of a client $c$ is located in $V_S$, the latency between them must be $t_1$.

If two clients both have their nearest servers in $V_C$, based on (a), their interaction path length would be $0+t_4+0 = t_4 > 2t_1 + t_2$. If two clients have their nearest servers in $V_C$ and $V_S$ respectively, the inter-server latency must be $t_3$ since the two servers map onto different elements. As a result, based on (a) and (b), the interaction path length between the two clients is $0 + t_3 + t_1 > (t_1 + t_2) + t_1 = 2t_1 + t_2$. Therefore, all the clients must have their nearest servers in $V_S$ in any valid solution of instance $\mathcal{T}$. Let $S \subseteq V_S$ be a valid solution. Without loss of generality, assume that each server in $S$ is connected by at least one client (any server not connected by any clients can simply be removed from $S$). According to (b), to cap the maximum interaction path length by $2t_1 + t_2$, the latency between any two distinct servers in $S$ must be bounded by $(2t_1 + t_2) - t_1 - t_1 = t_2$. Our assumption $t_4 > 2t_1 + t_2$ implies that $t_4 > t_2$. Thus, servers located in the same group of $V_S$ must come from the same clique therein. Since there are $k$ groups in $V_S$, the number of cliques with servers placed is at most $k$. Therefore, the subsets in $\mathbb{Q}$ that correspond to the cliques with servers placed form a set cover of size at most $k$ for instance $\mathcal{R}$.

The above analysis establishes that a set cover of size at most $k$ can be found for instance $\mathcal{R}$ if and only if there exists a valid solution for instance $\mathcal{T}$. Therefore, server provisioning on networks without the triangle inequality is NP-hard.

**Theorem 1.** *If P≠NP, no polynomial-time algorithm can achieve any constant approximation ratio for server provisioning on networks without the triangle inequality.*

*Proof:* This result can be proved based on the server provisioning instance constructed above. If there exists a set cover of size at most $k$, the maximum interaction path length under the optimal server placement is $2t_1 + t_2$. According to the above analysis, in a non-optimal server placement, either (1) some client and its nearest server do not map onto the same element, or (2) some client has its nearest server located in $V_C$, or (3) two servers in $V_S$ have latency $t_4$ to each other. In case (1), the interaction path from that client to itself is $2t_3$ or $2t_4$ long. In case (2), there exists an interaction path with length $t_4$ or $t_3+t_1$. In case (3), there exists an interaction path with length at least $t_4$. Since $t_3 > t_1 + t_2 > t_1$, the maximum interaction path length under a non-optimal server placement is at least $\min\{2t_3, 2t_4, t_4, t_3 + t_1\} = \min\{t_4, t_3 + t_1\}$. If a polynomial-time server provisioning algorithm achieves a constant approximation ratio $\gamma$, by setting $t_3 = t_4 = (2t_1 + t_2) \cdot \gamma$ in the constructed network, it is guaranteed to find a set cover of size at most $k$ if one exists, which contradicts P≠NP. Thus, server provisioning for networks without the triangle inequality is not approximable within any constant factor. ■

### *B. Hardness for Restricted Choices of Server Locations*

In the case that the choices of server locations are restricted to a subset of nodes in the network, the server provisioning problem is also NP-hard. Since server provisioning for networks without the triangle inequality has been shown to be NP-hard in the previous section, we shall focus our discussion here on server provisioning with restricted choices of server locations on networks with the triangle inequality only.

The NP-hardness can be proved by a similar reduction from the minimum set cover problem to that in the previous section. Given an instance $\mathcal{R}$ of the set cover problem, we construct the same network as shown in Figure 2. Suppose that the latency values $t_1$, $t_2$, $t_3$ and $t_4$ satisfy the triangle inequality for the constructed network. In addition, we also assume that

- $2t_3 > 2t_1 + t_2$,
- $t_4 > t_2$.

We restrict the choices of server locations to the nodes in $V_S$ only and define an instance $\mathcal{X}$ of the server provisioning problem as follows: assuming that a client is located at each node of $V_C$, can we select a set of server locations from $V_S$ to bound the maximum interaction path length by $2t_1 + t_2$?

It can be proved in the same way as in the previous section that a valid solution for the server provisioning instance $\mathcal{X}$ can always be derived from a set cover of size at most $k$. Next, we show that a set cover of size at most $k$ can always be derived from a valid solution for instance $\mathcal{X}$.

Note that the latency from any node in $V_C$ to any node in $V_S$ is either $t_1$ or $t_3$. If any client has latency exceeding $t_1$ to its nearest server, it must have a latency of $t_3$ to its nearest server. In consequence, the interaction path length from such client to itself is $2t_3 > 2t_1+t_2$. Thus, in any valid solution for instance $\mathcal{X}$, the latency from each client to its nearest server can only be $t_1$. This implies that each client and its nearest server should map onto the same element in set $P$ of the set cover instance $\mathcal{R}$. Consider a valid server placement $S$ in which each server is connected by at least one client. To cap the maximum interaction path length by $2t_1 + t_2$, the latency between any two distinct servers in $S$ must be bounded by $(2t_1 + t_2) - t_1 - t_1 = t_2$. Since $t_4 > t_2$, the servers located in the same group of $V_S$ must come from the same clique therein. Since there are $k$ groups in $V_S$, the number of cliques with servers placed is at most $k$. Therefore, the subsets in $\mathbb{Q}$

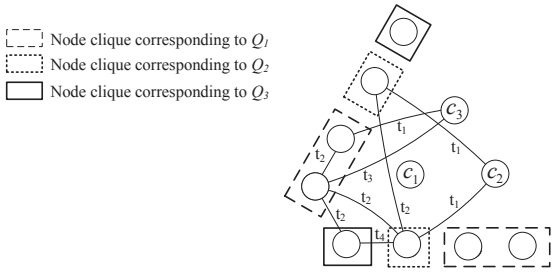



Fig. 3. Some triangles in the constructed network for server provisioning instance $\mathcal{X}$.

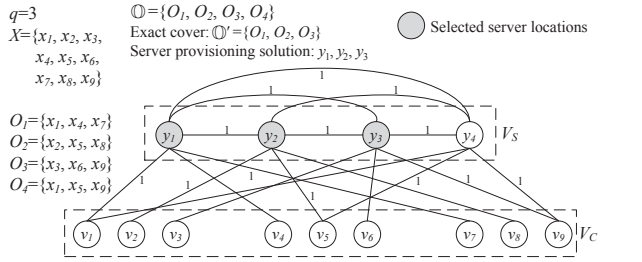



Fig. 4. Example instances of the X3C problem and the server provisioning problem for continuous DIAs

that correspond to the cliques with servers placed form a set cover of size at most $k$ for instance $\mathcal{R}$.

The above reduction implies that server provisioning with restricted choices of server locations on networks with the triangle inequality is NP-hard. Next, we analyze the non-approximability for server provisioning in this case.

**Theorem 2.** *If P≠NP, no polynomial-time algorithm can achieve an approximation ratio within* $3/2$ *for server provisioning with restricted choices of server locations on networks with the triangle inequality.*

*Proof:* Consider the server provisioning instance constructed above. According to the above analysis, if there exists a set cover of size at most $k$, the maximum interaction path length under the optimal server placement is $2t_1 + t_2$. On the other hand, the maximum interaction path length is at least $\min\{2t_3, 2t_1+t_4\}$ under a non-optimal server placement. This is because in a non-optimal server placement, either some client and its nearest server do not map onto the same element in set $P$, or the latency between some pair of servers is $t_4$. In the former case, that client has latency $t_3$ to its nearest server, so the interaction path from that client to itself is $2t_3$ long. In the latter case, there exists an interaction path with length at least $t_1 + t_4 + t_1 = 2t_1 + t_4$. Thus, no server placement can produce a maximum interaction path length between $2t_1 + t_2$ and $\min\{2t_3, 2t_1+t_4\}$. Therefore, if a polynomial-time server provisioning algorithm is able to achieve an approximation ratio within $\frac{\min\{2t_3, 2t_1+t_4\}}{2t_1+t_2}$, it is guaranteed to find a set cover of size at most $k$ if there exists one. This contradicts to the assumption of P≠NP since the set cover problem is NP-hard. Thus, server provisioning with restricted choices of server locations is not approximable within a ratio of $\frac{\min\{2t_3, 2t_1+t_4\}}{2t_1+t_2}$.

To yield the strongest non-approximability result, the latency values $t_1$, $t_2$, $t_3$ and $t_4$ can be tuned to make the ratio $\frac{\min\{2t_3, 2t_1+t_4\}}{2t_1+t_2}$ as large as possible. Note that there exist a triangle with sides $t_1$, $t_2$ and $t_3$ and another triangle with sides $t_2$, $t_2$ and $t_4$ in the constructed network (see Figure 3). By the triangle inequality, we have $2t_3 \leq 2t_1 + 2t_2$ and $t_4 \leq 2t_2$. Thus,

$$\frac{\min\{2t_3, 2t_1 + t_4\}}{2t_1 + t_2} \leq \frac{2t_1 + 2t_2}{2t_1 + t_2} = 1 + \frac{t_2}{2t_1 + t_2}.$$

Furthermore, the presence of a triangle with sides $t_1$, $t_1$ and $t_2$ in the constructed network (see Figure 3) implies that $t_2 \leq 2t_1$. Therefore, $t_2 \leq \frac{1}{2}(2t_1 + t_2)$. It follows that

$$\frac{\min\{2t_3, 2t_1 + t_4\}}{2t_1 + t_2} \leq 1 + \frac{t_2}{2t_1 + t_2} \leq 1 + \frac{1}{2} = \frac{3}{2}.$$

So, the largest possible ratio of $\frac{\min\{2t_3, 2t_1+t_4\}}{2t_1+t_2}$ is $\frac{3}{2}$. This can be achieved by setting $t_1 = 1$, $t_2 = 2$, $t_3 = 3$ and $t_4 = 4$. It is easy to verify that such latency setting satisfies the triangle inequality for the constructed network. Therefore, server provisioning with restricted choices of server locations on networks with the triangle inequality cannot be approximated with a ratio of $\frac{3}{2}$. ■

### C. Hardness for Limited Number of Server Locations to Select

When there is a limit on the number of server locations to select, the server provisioning problem is again NP-hard even if the network latency satisfies the triangle inequality, and there is no restriction on the choices of server locations. This can be proved by a polynomial reduction from the Exact Cover By 3-Sets (X3C) problem [20]. Given a set $X$ of $3q$ elements, and a collection $\mathbb{O}$ of $X$'s 3-element subsets, the X3C problem is to decide whether $\mathbb{O}$ contains an exact cover for $X$, i.e., a subcollection $\mathbb{O}' \subseteq \mathbb{O}$ such that every element of $X$ occurs in exactly one subset in $\mathbb{O}'$.

Given an instance $\mathcal{U}$ of the X3C problem, we first construct a network composed of node sets $V_C$ and $V_S$ as shown in Figure 4. $V_C$ is a set of $3q$ nodes, each corresponding to an element in set $X$. $V_S$ is a set of $|\mathbb{O}|$ nodes, each corresponding to a subset in $\mathbb{O}$. The nodes in $V_S$ are connected to each other by links of latency 1, and each node in $V_S$ has links of latency 1 to the three nodes in $V_C$ that represent the three elements of its corresponding subset. Unmarked node pairs have network latencies given by the lengths of shortest paths. Apparently, the network latencies satisfy the triangle inequality.

An instance $\mathcal{F}$ of the server provisioning problem is then defined as follows: Given the constructed network, assuming that a client is located at each node in $V_C$, and all the nodes in the network are candidate server locations, can we select up to $q$ locations to place servers such that the maximum interaction path length among all clients is bounded by 3?

First, we show that if there is an exact cover $\mathbb{O}' \subseteq \mathbb{O}$ for instance $\mathcal{U}$, a valid solution must exist for instance $\mathcal{F}$. Since each subset in $\mathbb{O}$ has exactly 3 elements, it is obvious that $|\mathbb{O}'| = q$. Let $S$ be the $q$ nodes in $V_S$ that represent the subsets in $\mathbb{O}'$. We select these nodes as server locations (see Figure 4 for an example). Since $\mathbb{O}'$ is an exact cover, each client has

exactly one link of latency 1 to a server in $S$ and connects to that server. Since all the nodes in $V_S$ are interconnected with each other by links of latency 1, the maximum interaction path length among all clients is bounded by $1+1+1=3$. Thus, placement $S$ is a valid solution for instance $\mathcal{F}$.

On the other hand, if we have a valid solution for instance $\mathcal{F}$, an exact cover for instance $\mathcal{U}$ can be found accordingly. We first show that selecting any nodes in $V_C$ to place servers would make the maximum interaction path length exceed the bound of 3. Suppose that $b$ ($b>0$) nodes in $V_C$ are selected as server locations. Then, the clients located at these nodes would connect to their co-located servers. The other clients have latency at least 2 to these servers. Among the nodes in $V_S$, at most $(q-b)$ of them could be selected to place servers. Since each node in $V_S$ has latency 1 to exactly three nodes in $V_C$, at most $3(q-b)$ clients can have latency 1 to the servers placed in $V_S$. The remaining $3q-3(q-b)-b=2b$ clients would have to connect to servers with latency at least 2 from them. So, the interaction path lengths among these $2b$ clients are at least $2+2>3$. Therefore, in a valid solution for instance $\mathcal{F}$, all the servers must be placed in $V_S$. In order to cap the maximum interaction path length at 3, each client must have latency 1 to its nearest server. Since there are $3q$ clients in total, $q$ server locations must be selected in $V_S$ and their corresponding 3-element subsets form an exact cover.

The above reduction shows that server provisioning with a limit on the number of server locations to select is NP-hard.

**Theorem 3.** *If P≠NP, no polynomial-time algorithm can achieve an approximation ratio within $4/3$ for server provisioning with a limit on the number of server locations to select.*

*Proof:* Consider the server provisioning instance $\mathcal{F}$ constructed above. If there exists an exact cover, an optimal server placement has a maximum interaction path length 3. On the other hand, there always exist clients having latency at least 2 to their nearest servers in non-optimal placements, giving rise to interaction paths of length at least 4. This implies that no server placement can produce a maximum interaction path length between 3 and 4. Therefore, if a polynomial-time algorithm can achieve an approximation ratio within $4/3$, it can always find a server placement with maximum interaction path length 3 if there exists one, following which an exact cover can be derived. Since the X3C problem is NP-hard, this leads to contradiction to the assumption P≠NP. ■

In following sections, we present and analyze several server placements for the server provisioning problem. The network latencies between clients and candidate server locations are needed for the computation of these algorithms. They can be acquired with existing tools like ping and King [21]. We start by analyzing the classical $k$-center server placement. We show that, in networks with the triangle inequality, the $k$-center server placement has an approximation ratio of 3 for our server provisioning problem. We then propose an efficient M-GREEDY algorithm that achieves an approximation ratio of 2. Finally, an algorithm called M-BETTER is proposed to further improve the approximation ratio to $\frac{5}{3}$.

## IV. $k$-CENTER SERVER PLACEMENT

Our server provisioning problem bears some similarity to the classical $k$-center problem in that both problems aim to find server placement that minimizes a maximum latency metric in a network of clients and servers. However, different from our problem, the $k$-center problem targets at placing a given number of $k$ servers in the network to minimize the maximum latency between the clients and their nearest servers. It does not consider the latencies among the servers. When a limit $k$ is set on the number of server locations to select in our server provisioning problem, we have the following approximability result of the $k$-center server placement.

**Theorem 4.** *For networks with the triangle inequality, the $k$-center server placement has an approximation ratio of 3 for the server provisioning problem with a limit $k$ on the number of server locations to select.*

*Proof:* Let $C$ be the set of clients. Suppose that the sets of optimal server locations for our server provisioning problem and the $k$-center problem are $S_O$ and $S_R$ respectively. For each client $c\in C$, denote by $o(c)$ and $r(c)$ the nearest servers to $c$ under placements $S_O$ and $S_R$ respectively. Then, for any two clients $c_i$ and $c_j\in C$, their interaction path length under placement $S_R$ is given by

$$l_R(c_i,c_j)=d(c_i,r(c_i))+d(r(c_i),r(c_j))+d(r(c_j),c_j).$$

By the triangle inequality, we have

$$\begin{aligned} l_R(c_i,c_j) &\le d(c_i,r(c_i))+\Big(d(r(c_i),c_i)+d(c_i,c_j)\\ &\quad +d(c_j,r(c_j))\Big)+d(r(c_j),c_j)\\ &= 2d(c_i,r(c_i))+2d(c_j,r(c_j))+d(c_i,c_j). \end{aligned}$$

Let $m=\max_{c\in C} d(c,r(c))$ be the maximum latency from the clients to their nearest servers under the $k$-center placement $S_R$. It follows that

$$l_R(c_i,c_j)\le 4m+d(c_i,c_j).$$

By the definition of the $k$-center placement, $m$ is the minimum achievable value of the maximum latency from the clients to their nearest servers among all possible placements of $k$ servers and hence among all possible placements of up to $k$ servers. Thus, there must exist a client $c_x\in C$ satisfying $d(c_x,o(c_x))\ge m$ under the optimal placement $S_O$. Therefore, by the triangle inequality, we have

$$\begin{aligned} l_R(c_i,c_j) &\le 2\cdot\big(2\cdot d(c_x,o(c_x))\big)+d(c_i,c_j)\\ &\le 2\cdot\Big(d(c_x,o(c_x))+d(o(c_x),o(c_x))+d(o(c_x),c_x)\Big)\\ &\quad +\Big(d(c_i,o(c_i))+d(o(c_i),o(c_j))+d(o(c_j),c_j)\Big). \end{aligned}$$

Denote by $L_O$ the maximum interaction path length under the optimal server placement $S_O$. Then,

$$d(c_x,o(c_x))+d(o(c_x),o(c_x))+d(o(c_x),c_x)\le L_O,$$

and

$$d(c_i,o(c_i))+d(o(c_i),o(c_j))+d(o(c_j),c_j)\le L_O.$$

It follows that

$$l_R(c_i, c_j) \leq 3 \cdot L_O. \quad (1)$$

The above inequality holds for any two clients $c_i$ and $c_j$. As a result, the maximum interaction path length $L_R$ under the $k$-center server placement satisfies

$$L_R = \max_{c_i, c_j \in C} l_R(c_i, c_j) \leq 3 \cdot L_O.$$

■

The approximation ratio 3 of the $k$-center server placement is tight. A tight example is given in Figure 5, which consists of four candidate server locations $\{s_1, s_2, o_1, o_2\}$ and two clients $\{c_1, c_2\}$. The latencies between unmarked node pairs are given by the lengths of their shortest paths. Suppose that $\varepsilon > 0$ and up to 2 server locations can be selected. Then, the set of optimal server locations for our server provisioning problem is $\{o_1, o_2\}$. Under such placement, clients $c_1$ and $c_2$ are connected to servers $o_1$ and $o_2$ respectively, so the maximum interaction path length is $2+3\varepsilon$. On the other hand, the 2-center placement is $\{s_1, s_2\}$, which gives a maximum latency of 1 between the clients and their nearest servers. In this case, the maximum interaction path length is $6+3\varepsilon$. Thus, the ratio between the two results is $\frac{6+3\varepsilon}{2+3\varepsilon}$, which can be made arbitrarily close to $\lim_{\varepsilon \to 0} \frac{6+3\varepsilon}{2+3\varepsilon} = 3$ when $\varepsilon$ approaches 0.

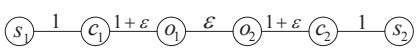



Fig. 5. The approximation ratio 3 of the $k$-center placement is tight.

## V. M-GREEDY SERVER PLACEMENT

The $k$-center placement is difficult to find as the $k$-center problem itself is also NP-hard [20]. In this section, we develop an efficient heuristic called M-GREEDY that achieves an even better approximation ratio than the $k$-center placement.

Algorithm 1 shows the pseudo code of the M-GREEDY algorithm. Denote by $S$ the set of server locations selected. Starting from an empty set $S$, the algorithm iteratively adds a candidate server location $s^* \in Z \setminus S$ to $S$ in a greedy manner that results in the largest reduction in the maximum interaction path length $M$ (lines 12, 13 and 18). If adding any server location from $Z \setminus S$ cannot further shorten the maximum interaction path length (lines 16-17), or when $|S|$ reaches a given limit $k$ (if any) on the number of server locations to select (line 7), the algorithm terminates and outputs the final server placement $S$ (line 27).

A brute force calculation of the maximum interaction path length has a time complexity of $O(|C|^2)$, where $|C|$ is the number of clients. Thus, a naive implementation of the above greedy algorithm has a computational complexity of $O(|Z| \cdot |C|^2)$ for each iteration and $O(k \cdot |Z| \cdot |C|^2)$ in total. To efficiently compute the maximum interaction path length when testing and selecting new server locations, we decompose the maximum length of the interaction paths involving each server $s$ into two parts: a local latency $L(s)$ and a remote latency $R(s)$. The local latency is defined as that from $s$ to the furthest client connecting to it, i.e.,

$$L(s) = \max_{n(c,S)=s} d(c, s),$$

```
1:  S ← ∅; //server locations selected
2:  M_p ← ∞; //maximum interaction path length for placement S
3:  for all z ∈ Z do
4:      ℂ(z) ← C; R_p(z) ← 0; L_p(z) ← 0; E_p(z) ← ∅;
5:  for all c ∈ C do
6:      n(c) ← ∅;
7:  for i = 1 to k do
8:      M* ← ∞, s* ← ∅;
9:      for all z ∈ Z \ S do
10:         updateLR(z); //invoke Algorithm 2
11:         M ← max_{s∈S∪{z}} {L(s) + R(s)};
12:         if M < M* then
13:             M* ← M, s* ← z;
14:             for all s ∈ S ∪ {z} do
15:                 R*(s) ← R(s); L*(s) ← L(s); E*(s) ← E(s);
16:     if M* ≥ M_p then
17:         break;
18:     S ← S ∪ {s*}; M_p ← M*;
19:     for all c ∈ ℂ(s*) do
20:         ℂ(n(c)) ← ℂ(n(c)) \ {c}; n(c) ← s*;
21:     for all z ∈ Z \ S do
22:         for all c ∈ ℂ(z) do
23:             if d(c, z) > d(c, s*) then
24:                 ℂ(z) ← ℂ(z) \ {c};
25:     for all s ∈ S do
26:         R_p(s) ← R*(s); L_p(s) ← L*(s); E_p(s) ← E*(s);
27: return S;
```

**Algorithm 1:** M-GREEDY server provisioning algorithm

where $n(c, S)$ denotes client $c$'s nearest server in $S$. The remote latency $R(s)$, on the other hand, is defined as the maximum latency from $s$ to all the clients through their connected servers, i.e.,

$$R(s) = \max_{c \in C} \big\{ d(c, n(c, S)) + d(n(c, S), s) \big\}.$$

In essence, $R(s)$ represents the longest possible latency for server $s$ to receive an operation issued by a client, and $L(s)$ represents the longest possible latency for $s$ to deliver a state update to its connected clients. The maximum interaction path length among all clients can be rewritten as

$$\begin{aligned}
&\max_{c,c' \in C} \big\{ d(c, n(c, S)) + d(n(c, S), n(c', S)) + d(n(c', S), c') \big\} \\
&= \max_{c \in C} \Big\{ d(c, n(c, S)) + \max_{c' \in C} \big\{ d(n(c, S), n(c', S)) \\
&\quad + d(n(c', S), c') \big\} \Big\} \\
&= \max_{s \in S} \Big\{ \max_{n(c,S)=s} \{ d(c, s) \} \\
&\quad + \max_{c' \in C} \big\{ d(s, n(c', S)) + d(n(c', S), c') \big\} \Big\} \\
&= \max_{s \in S} \big\{ L(s) + R(s) \big\}.
\end{aligned}$$

The M-GREEDY algorithm makes use of the above decomposition in computing the maximum interaction path length (line 11). The local and remote latencies of all servers are incrementally updated across iterations. In addition, the algorithm also maintains, for each client $c$, its nearest server $n(c)$ under the current server placement $S$; for each selected server location $s \in S$, the set of clients $\mathbb{C}(s)$ whose nearest servers are $s$ under placement $S$; and for each unselected candidate server location $z \in Z \setminus S$, the set of clients $\mathbb{C}(z)$

updateLR($z$)
1: $\Delta S \leftarrow \{n(c) \mid c \in \mathbb{C}(z)\}$;
2: **for all** $s \in S \setminus \Delta S$ **do**
3: $L(s) \leftarrow L_p(s)$;
4: **for all** $s \in \Delta S$ **do**
5: $L(s) \leftarrow \max_{c \in \mathbb{C}(s) \setminus \mathbb{C}(z)} d(c, s)$;
6: **if** $L(s) = L_p(s)$ **then**
7: $\Delta S \leftarrow \Delta S \setminus \{s\}$;
8: $L(z) \leftarrow \max_{c \in \mathbb{C}(z)} d(c, z)$;
9: **for all** $s \in S$ **do**
10: **if** $d(s, z) + L(z) > R_p(s)$ **then**
11: $R(s) \leftarrow d(s, z) + L(z)$; $E(s) \leftarrow \{z\}$;
12: **else if** $E_p(s) \setminus \Delta S \neq \emptyset$ **then**
13: $R(s) \leftarrow R_p(s)$; $E(s) \leftarrow E_p(s) \setminus \Delta S$;
14: **if** $d(s, z) + L(z) = R_p(s)$ **then**
15: $E(s) \leftarrow E(s) \cup \{z\}$;
16: **else**
17: $R(s) \leftarrow \max_{x \in S \cup \{z\}} \{d(s, x) + L(x)\}$;
18: $E(s) \leftarrow \{x \in S \cup \{z\} \mid d(s, x) + L(x) = R(s)\}$;
19: $R(z) \leftarrow \max_{x \in S \cup \{z\}} \{d(z, x) + L(x)\}$;
20: $E(z) \leftarrow \{x \in S \cup \{z\} \mid d(z, x) + L(x) = R(z)\}$;

**Algorithm 2:** Update local and remote latencies

whose nearest servers would become $z$ if $z$ is added to the current placement $S$. Initially, $n(c) = \varnothing$ for all clients (line 6) and $\mathbb{C}(z) = C$ (the whole client set) for all candidate server locations (line 4). Since the clients always connect to their nearest servers, if a candidate server location $s^*$ is added to the current placement $S$, for each client $c \in \mathbb{C}(s^*)$, $c$ shall no longer connect to its current server $n(c)$ and is thus removed from $\mathbb{C}(n(c))$ (lines 19-20). Meanwhile, for each remaining candidate location $z \in Z \setminus S$, all the clients $c$ satisfying $d(c, z) > d(c, s^*)$ shall be removed from $\mathbb{C}(z)$ as it would no longer be possible for them to connect to $z$ even if $z$ is selected later (lines 21-24).

Algorithm 2 shows the subroutine for evaluating the change in the local and remote latencies caused by adding a new server location $z$. To update the local latencies, we start by finding the set of existing servers $\Delta S$ which have at least one client that will re-connect to $z$ if $z$ is selected as a server location (line 1). Only the local latencies of the servers in $\Delta S$ need to be recalculated (lines 4-5). Then, we remove from $\Delta S$ the servers whose local latencies are not changed (lines 6-7). So, $\Delta S$ eventually constitutes the servers whose local latencies change due to the addition of $z$. It is obvious that the local latencies of the servers in $\Delta S$ can only decrease after $z$ is added. Line 8 calculates $z$'s own local latency. The update of remote latencies leverages the values of local latencies. Note that the remote latency $R(s)$ of each server $s$ can be rewritten as

$$R(s) = \max_{x \in S} \big\{d(s, x) + L(x)\big\}.$$

For each server $s$, we also maintain the list of servers $E(s)$ which give rise to the maximum remote latency of $s$, i.e.,

$$E(s) = \big\{x \in S \mid d(s, x) + L(x) = R(s)\big\}.$$

Let $L_p(s)$, $R_p(s)$ and $E_p(s)$ be the information of each server $s$ before $z$ is added, and let $L(s)$ be the updated local latency of $s$ after $z$ is added. Since the local latencies of existing servers cannot increase due to the addition of a new server, after $z$ is added, the remote latency of each existing server $s \in S$ is bounded by $\max\{d(s, z) + L(z), R_p(s)\}$. If $d(s, z) + L(z) > R_p(s)$, the $R(s)$ and $E(s)$ after adding $z$ are simply given by $d(s, z) + L(z)$ and $\{z\}$ respectively (lines 10-11). Otherwise, if $E_p(s) \setminus \Delta S \neq \emptyset$, the remote latency of $s$ remains unchanged after the addition of $z$ (lines 12-13). In this case, $z$ needs to be added to $E(s)$ if $d(s, z) + L(z) = R_p(s)$ (i.e., $z$ also gives rise to the remote latency of $s$) (lines 14-15). If $E_p(s) \setminus \Delta S = \emptyset$, the remote latency of $s$ is recalculated as $R(s) = \max_{x \in S \cup \{z\}} \{d(s, x) + L(x)\}$ (line 17). Finally, lines 19-20 calculate $z$'s own remote latency. The subroutine of Algorithm 2 has a time complexity of $O(|C| + k^2)$, where $O(|C|)$ is the total computational complexity of lines 4-5 for all selected server locations $s$, $O(k^2)$ is that of line 17 for all selected server locations $s$, and $k$ is the number of iterations in Algorithm 1. Therefore, the total time complexity of the M-GREEDY algorithm is $O(k \cdot |Z| \cdot (|C| + k^2))$.

**Theorem 5.** *For networks with the triangle inequality, the* M-GREEDY *server placement has an approximation ratio of* 2 *for the server provisioning problem.*

*Proof:* When only one server is placed in the network, there is no inter-server latency involved in the interactions between clients. In this case, the longest interaction path is the one from the client furthest away from the server to itself.

Suppose that $s^*$ is the first server location selected by the M-GREEDY algorithm. Let $c_i$ be the client furthest away from $s^*$. Then, after selecting $s^*$, the maximum interaction path length is $2 \cdot d(c_i, s^*)$. Denote by $o(c_i)$ the nearest server of $c_i$ in an optimal server placement $S_O$. There must exist a client $c_j$ (which can be the same as $c_i$) satisfying

$$d(c_j, o(c_i)) \geq d(c_i, s^*) = \max_{c \in C} d(c, s^*).$$

This is because otherwise, $o(c_i)$ would have been the first server location selected by the M-GREEDY algorithm. Denote by $o(c_j)$ the nearest server of $c_j$ in the optimal server placement $S_O$. Let $L_O$ be the maximum interaction path length in the optimal server placement. Then, by the triangle inequality,

$$\begin{aligned} L_O &\geq d(c_j, o(c_j)) + d(o(c_j), o(c_i)) + d(o(c_i), c_i) \\ &\geq d(c_j, o(c_i)) + d(o(c_i), c_i) \\ &\geq d(c_j, o(c_i)). \end{aligned}$$

Thus,

$$2d(c_i, s^*) \leq 2d(c_j, o(c_i)) \leq 2 \cdot L_O.$$

Since the maximum interaction path length finally produced by the M-GREEDY algorithm cannot be longer than $2d(c_i, s^*)$ (that after selecting the first location $s^*$), it must also be within two times of that in $S_O$. ■

The approximation ratio 2 of the M-GREEDY algorithm holds regardless of whether the choices of server locations are restricted and whether there is a limit on the number of server locations to select. The approximation ratio is also tight. Figure 6 shows a tight example which consists of 5 nodes $\{c_1, c_2, c_3, c_4, s\}$. Suppose that a client is located at each of the nodes $c_1$, $c_2$, $c_3$ and $c_4$, and all the nodes in the network are candidate server locations. Then, the first server

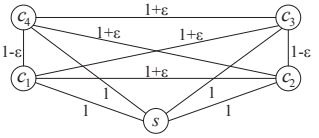



Fig. 6. The approximation ratio 2 of the M-GREEDY algorithm is tight.

location selected by the M-GREEDY algorithm is $s$, because the maximum interaction path length is 2 if all the clients are connected to $s$. If the first server is placed at any other location, the maximum interaction path length is $2 + 2\varepsilon > 2$. After $s$ is selected, further placing a new server at any node $c_i$ $(i = 1, 2, 3, 4)$ would result in some client located at $c_j \neq c_i$ to connect to the server at $c_i$ with latency $1 - \varepsilon$. As a result, the maximum interaction path length becomes $(1 - \varepsilon) + 1 + 1 = 3 - \varepsilon > 2$. Therefore, the M-GREEDY algorithm stops at the second iteration and outputs the server placement $\{s\}$. On the other hand, the optimal placement of this example is to place servers at nodes $\{c_1, c_2, c_3, c_4\}$. In this way, each client is connected to the server at the same location. Thus, the maximum interaction path length is $1 + \varepsilon$. When $\varepsilon$ approaches 0, the M-GREEDY algorithm and the optimal server placement differ by a ratio of $\lim_{\varepsilon \to 0} \frac{2}{1+\varepsilon} = 2$ in the maximum interaction path length.

## VI. M-BETTER SERVER PLACEMENT

In this section, we propose a M-BETTER algorithm that further improves the approximation ratio over M-GREEDY. We first define a simple server placement $S_N$ called NEAREST that selects the collection of all clients' nearest candidate server locations. Suppose that $C$ is the set of clients and $Z$ is the set of candidate server locations. For each client $c \in C$, let $n(c)$ be the nearest candidate server location to $c$, i.e., $d(c, n(c)) = \min_{s \in Z}\{d(c, s)\}$. Then, $S_N = \bigcup_{c \in C} n(c)$. The computational complexity of NEAREST is $O(|Z| \cdot |C|)$.

The M-BETTER algorithm applies a heuristic to combine the NEAREST server placement $S_N$ and the M-GREEDY server placement $S_G$. Suppose that $L_N$ and $L_G$ are the maximum interaction path lengths under server placements $S_N$ and $S_G$ respectively. The M-BETTER algorithm selects the candidate server locations in $S_N$ if $L_N < L_G$, and selects the candidate server locations in $S_G$ otherwise. In this way, the M-BETTER server placement produces a maximum interaction path length $L_B = \min\{L_N, L_G\}$. Since the M-GREEDY server placement has higher computational complexity than the NEAREST server placement, the time complexity of the M-BETTER algorithm is the same as that of the M-GREEDY algorithm.

**Theorem 6.** *For networks with the triangle inequality, the* M-BETTER *server placement has an approximation ratio of* $\frac{5}{3}$ *for the server provisioning problem.*

*Proof:* Let $S_N$ be the set of server locations selected by the NEAREST server placement. For each client $c \in C$, denote by $n(c) \in S_N$ the nearest server of $c$ under placement $S_N$. Then, $n(c)$ is the nearest candidate server location to $c$. Let $r = \max_{c \in C} d(c, n(c))$ be the maximum latency from all the clients to their nearest servers under server placement $S_N$. Denote by $L_N$ the maximum interaction path length under placement $S_N$. Suppose that $L_N$ is produced by the interaction path between two clients $c_x$ and $c_y \in C$. According to the triangle inequality, we have

$$\begin{aligned} L_N &= d(c_x, n(c_x)) + d(n(c_x), n(c_y)) + d(n(c_y), c_y) \\ &\leq d(c_x, n(c_x)) + \Big(d(n(c_x), c_x) + d(c_x, c_y) \\ &\quad + d(c_y, n(c_y))\Big) + d(n(c_y), c_y) \\ &= 2d(c_x, n(c_x)) + d(c_x, c_y) + 2d(c_y, n(c_y)) \\ &\leq 2r + d(c_x, c_y) + 2r. \end{aligned} \tag{2}$$

Suppose that $S_O \subseteq Z$ is a set of optimal server locations for the server provisioning problem. For each client $c \in C$, denote by $o(c) \in S_O$ the nearest server of $c$ under placement $S_O$. Then, it follows from (2) that

$$\begin{aligned} L_N &\leq 4r + \Big(d(c_x, o(c_x)) + d(o(c_x), o(c_y)) + d(o(c_y), c_y)\Big) \\ &\leq 4r + L_O, \end{aligned} \tag{3}$$

where $L_O$ is the maximum interaction path length under the optimal server placement $S_O$.

Next, consider the M-GREEDY server placement. Suppose that the first server location selected by the M-GREEDY algorithm is $s^* \in Z$. Let $t = \max_{c \in C} d(c, s^*)$ be the latency from $s^*$ to the furthest client. Then, the maximum interaction path length given by connecting all the clients to $s^*$ is $2t$. Note that the M-GREEDY algorithm can only improve the maximum interaction path length by selecting additional server locations. Thus, the maximum interaction path length $L_G$ eventually produced by M-GREEDY cannot be longer than $2t$, i.e.,

$$L_G \leq 2t. \tag{4}$$

Since the M-BETTER algorithm produces a maximum interaction path length $L_B = \min\{L_N, L_G\}$, it follows that

$$L_B \leq 2t. \tag{5}$$

By the design of the M-GREEDY algorithm, $t$ is the lowest achievable value of the maximum latency from all the clients to any single server. Therefore, for the nearest server $o(c_i)$ of each client $c_i$ under the optimal placement $S_O$, there must exist a client $c_j \in C$ (which can be the same as $c_i$) fulfilling $d(c_j, o(c_i)) \geq t$.

If there exists a client $c_i$ such that $d(c_i, o(c_i)) \geq t$, we have

$$L_O \geq 2d(c_i, o(c_i)) \geq 2t.$$

It follows from (5) that $L_O \geq L_B$. Thus, it must hold that $L_B = L_O$ since $L_O$ is the maximum interaction path length of an optimal server placement. This implies that M-BETTER must produce an optimal server placement in this case.

Otherwise, if all clients $c_i$ fulfill $d(c_i, o(c_i)) < t$, then for each client $c_i$, there must exist another client $c_j$ $(c_j \neq c_i)$ satisfying $d(c_j, o(c_i)) \geq t$. Consider the client $c_k$ that is furthest away from its nearest server under the optimal placement $S_O$, i.e., $d(c_k, o(c_k)) = \max_{c \in C} d(c, o(c))$. Let $c_h$ $(c_h \neq c_k)$ be a client satisfying $d(c_h, o(c_k)) \geq t$. Then, by the definition of $L_O$ and the triangle inequality, we have

$$L_O \geq d(c_h, o(c_h)) + d(o(c_h), o(c_k)) + d(o(c_k), c_k)$$

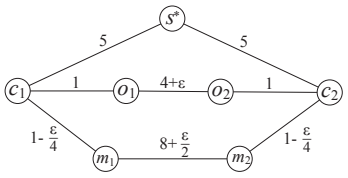



Fig. 7. The approximation ratio $\frac{5}{3}$ of the M-BETTER algorithm is tight.

$$\geq d(c_h, o(c_k)) + d(o(c_k), c_k)$$
$$\geq t + d(o(c_k), c_k).$$

Note that for each client $c$, $n(c)$ is the nearest candidate server location to $c$. Thus, $d(c, o(c)) \geq d(c, n(c))$. It follows that

$$d(c_k, o(c_k)) = \max_{c \in C} d(c, o(c)) \geq \max_{c \in C} d(c, n(c)) = r.$$

Therefore,

$$L_O \geq t + r.$$

According to (3), we have

$$r \geq \frac{1}{4} \cdot \big(L_N - L_O\big).$$

It follows that

$$L_O \geq t + \frac{1}{4} L_N - \frac{1}{4} L_O.$$

Thus,

$$L_O \geq \frac{4}{5} \cdot \Big(t + \frac{1}{4} L_N\Big).$$

Based on (4), we have

$$\begin{aligned} L_O &\geq \frac{4}{5} \cdot \Big(\frac{1}{2} L_G + \frac{1}{4} L_N\Big) \\ &= \frac{2}{5} \cdot L_G + \frac{1}{5} \cdot L_N \\ &\geq \frac{3}{5} \cdot \min\big\{L_N, L_G\big\} \\ &= \frac{3}{5} \cdot L_B. \end{aligned}$$

Therefore,

$$L_B \leq \frac{5}{3} \cdot L_O.$$

■

The approximation ratio $\frac{5}{3}$ of the M-BETTER algorithm is tight. Figure 7 illustrates a tight example. The latencies between unmarked node pairs are given by the lengths of their shortest paths. Suppose that two clients are located at nodes $c_1$ and $c_2$ respectively, and the remaining nodes in the network are all candidate server locations. Since each client $c_i$ ($i = 1$, 2) has a latency of $1 + (4 + \varepsilon) = 5 + \varepsilon > 5$ to node $o_j$ ($j \neq i$) and a latency of $(1 - \frac{\varepsilon}{4}) + (8 + \frac{\varepsilon}{2}) = 9 + \frac{\varepsilon}{4} > 5$ to node $m_j$ ($j \neq i$), the first server selected by the M-GREEDY algorithm is $s^*$. Connecting both clients to $s^*$ gives a maximum interaction path length of $5 + 5 = 10$. If one node $o_i$ or $m_i$ ($i = 1, 2$) is further selected, client $c_i$ would re-connect to the newly selected node, and this results in an interaction path length between $c_1$ and $c_2$ greater than 10. Therefore, the M-GREEDY algorithm terminates at the second iteration and produces the server placement $\{s^*\}$. On the other hand, the nearest candidate server locations of clients $c_1$ and $c_2$ are $m_1$ and $m_2$ respectively. So, the maximum interaction path length under the NEAREST server placement $\{m_1, m_2\}$ is $(1 - \frac{1}{4}\varepsilon) + (8 + \frac{1}{2}\varepsilon) + (1 - \frac{1}{4}\varepsilon) = 10$. Therefore, the M-BETTER algorithm also produces a maximum interaction path length of 10. In this example, the optimal server placement is to select nodes $o_1$ and $o_2$ as server locations. Under such server placement, clients $c_1$ and $c_2$ are connected to servers $o_1$ and $o_2$ respectively. As a result, the maximum interaction path length is $1 + (4 + \varepsilon) + 1 = 6 + \varepsilon$. When $\varepsilon$ approaches 0, the ratio between the maximum interaction path lengths produced by the M-BETTER heuristic and the optimal server placement is $\lim_{\varepsilon \to 0} \frac{10}{6+\varepsilon} = \frac{5}{3}$.

It is worth noting that the approximation ratio $\frac{5}{3}$ of the M-BETTER algorithm holds regardless of whether the choices of server locations are restricted. In the case of restricted choices, M-BETTER is near optimal in the sense that this ratio is quite close to the lower bound $\frac{3}{2}$ given by an aforementioned non-approximability result (Theorem 2). On the other hand, since the NEAREST algorithm does not consider any restriction on the number of server locations to select, neither is M-BETTER able to respect such restrictions. Thus, in the case where there is a limit on the number of server locations to select, M-GREEDY may have to be adopted for server provisioning.

## VII. EXPERIMENTAL EVALUATION

To conduct experimental evaluation, we developed a simulator written in C++ that takes network latency datasets as input to emulate underlying networks of DIAs. The experiments are carried out on an Intel Xeon 3.2GHz workstation with 16GB RAM. Our evaluations with different datasets including Meridian [22] and PlanetLab [23] have shown similar performance trends. Due to space limitations, we shall focus on presenting the experimental results for the Meridian dataset. The Meridian dataset is a real Internet latency dataset collected using the King technology [21]. To our knowledge, it is also the largest publicly available Internet latency dataset. The dataset provides the measurements of pairwise network latencies among 2500 nodes. Since some measurements are not available, we discard those nodes involved in the unavailable measurements and keep the remaining 1796 nodes that form a complete matrix of pairwise latencies.

We compare the M-GREEDY and M-BETTER server placements against other baseline placements. To quantify the performance difference of the algorithms, we normalize their maximum interaction path lengths by a theoretical lower bound. Given a set of candidate server locations $Z$, the shortest possible interaction path between two client $c_i$ and $c_j$ is $\min_{s_a, s_b \in Z} \big\{d(c_i, s_a) + d(s_a, s_b) + d(s_b, c_j)\big\}$. Thus, a lower bound on the maximum interaction path length is given by

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

The above bound makes two relaxations. First, a server is assumed to be available at every candidate server location. Second, each client can connect to different servers for interacting with different clients. Thus, this bound is a super-

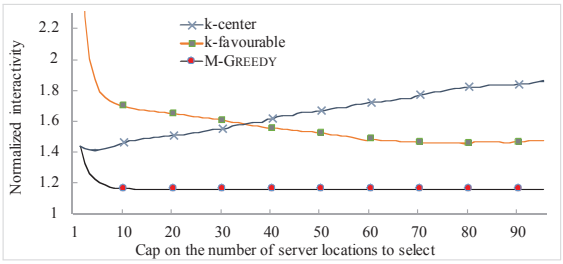



Fig. 8. Normalized interactivity of different algorithms when there are 300 candidate server locations

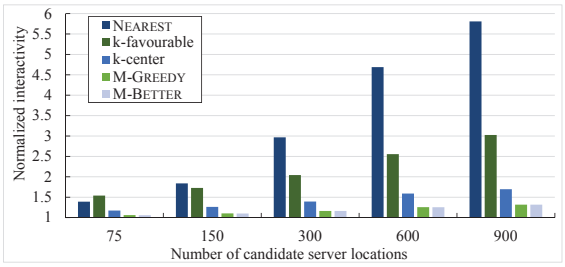



Fig. 9. Normalized interactivity of different algorithms for different numbers of candidate server locations.

optimum and may not really be achievable. We refer to the normalized maximum interaction path length as the *normalized interactivity*.

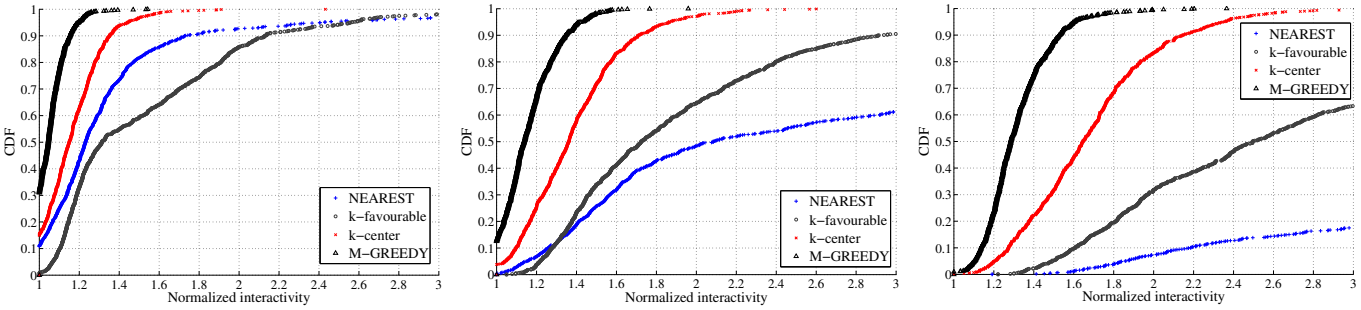
The NP-hardness of the $k$-center problem indicates that it is difficult to find the exact $k$-center server placement. For comparison purpose, we implement a $k$-center heuristic [24] in our experiments. Starting from an empty placement, the $k$-center heuristic iteratively selects a new server location that leads to the largest reduction in the maximum latency from the clients to their nearest servers until exactly $k$ servers are placed. In addition, we also compare our algorithms against the NEAREST server placement introduced in Section VI and a $k$-favourable heuristic. The idea of the $k$-favourable heuristic is developed from the above derivation of the lower bound. It evaluates the popularity of each candidate server location by the frequency of its involvement in the shortest possible interaction paths among all client pairs. Then, the $k$-favourable heuristic selects the $k$ most popular candidate server locations.

### A. Performance Comparison for Different Algorithms

Our experiments assume that the clients and candidate server locations are separate. For each simulation run, we randomly select a total of 896 nodes and assume that a client is located at each of these nodes. Then, 75, 150, 300, 600 or 900 nodes are randomly selected from the remaining nodes as the candidate server locations. For each setting, we perform 1000 simulation runs using different sets of clients and candidate server locations. The results of various simulation runs are normalized by different lower bound values derived using the respective sets of candidate server locations.

We first study the maximum interaction path lengths of different algorithms as a function of the limit on the number of server locations to select. The number of candidate server locations is set to 300 in this experiment. Figure 8 shows the average results of 1000 simulation runs for the $k$-center, $k$-favourable and M-GREEDY server placements. Since the NEAREST algorithm and the M-BETTER algorithm do not consider any limitation on the number of server locations to select, they are not included in this experiment. Note that the first server location selected by our M-GREEDY algorithm constitutes the 1-center placement. Starting from the 1-center placement, the M-GREEDY algorithm considerably reduces the maximum interaction path length by adding more server locations. This implies that there is significant potential for improving the interactivity performance by distributing the servers hosting continuous DIAs. However, increasing the number of server locations does not guarantee to enhance the interactivity unless the server locations are carefully chosen. For example, it can be seen from Figure 8 that the performance of the $k$-center placement deteriorates when the number of server locations selected becomes large. This is because the $k$-center placement does not take into account the inter-server latencies, which may increase substantially and outweigh the reduction in the latencies from clients to servers when the server locations are more widely dispersed. By considering the client-to-server latencies and the inter-server latencies together, the $k$-favourable heuristic is able to cut the maximum interaction path length as more server locations are selected. However, its performance is still far worse than the M-GREEDY algorithm. The best achievable result of the M-GREEDY algorithm is 1.16, indicating that it can produce near-optimal server placements.

Next, we study the impact of the number of candidate server locations. We remove the limit on the number of server locations to select and run our M-GREEDY algorithm until it terminates due to possible performance deterioration if more servers are placed. We record the exact number of server locations selected in each simulation run. We then use the recorded number as the number of server locations to pick for the $k$-center and $k$-favourable server placements in the same simulation run. This leads to exactly the same number of server locations selected by these three algorithms for fair comparison. The NEAREST and M-BETTER server placements do not have any stipulation on the number of server locations to select. Figure 9 shows the average results of 1000 simulation runs. It can be seen that the interactivity performance achieved by the M-BETTER server placement is very close to that of the M-GREEDY server placement. This is because our M-GREEDY algorithm produces better solutions than the NEAREST server placement in most of the simulation runs. The performance improvement of the M-GREEDY and M-BETTER server placements relative to the other three placements increases with the number of candidate server locations. This implies that M-GREEDY and M-BETTER are effective in finding good server placements even when the search space is large. It can also be seen from Figure 9 that the performance of the NEAREST server placement deteriorates sharply with increasing number of candidate server locations. This indicates that aggressively shortening the client-to-server latencies cannot contribute effectively to




(a) 75 candidate server locations (b) 300 candidate server locations (c) 900 candidate server locations

Fig. 10. Cumulative distribution of normalized interactivity for 1000 simulation runs

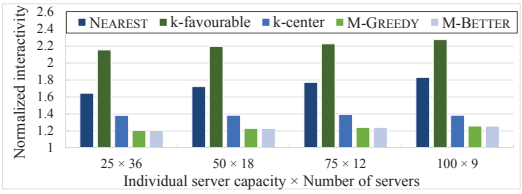



Fig. 11. Normalized interactivity of different algorithms for different individual server capacities.

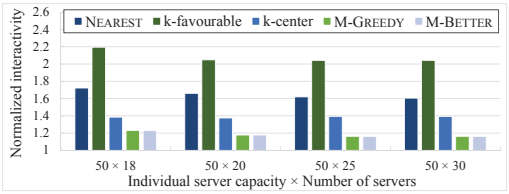



Fig. 12. Normalized interactivity of different algorithms for different total server capacities.

improving the interactivity performance of continuous DIAs.

Figure 10 shows the cumulative distribution of the normalized interactivity for the 1000 simulation runs of different numbers of candidate server locations. For ease of reading, the M-BETTER server placement is not shown in the figure since it produces almost the same curve as the M-GREEDY server placement. When there are only 75 candidate server locations (Figure 10(a)), our M-GREEDY algorithm finds the optimal server placement in over 30% of the simulation runs (a normalized interactivity of 1 implies that the output placement must be optimal), whereas the $k$-center placement finds it in just 15% of the runs and the other two placements find even less. Our detailed investigation indicates that most of the simulation runs in which the $k$-center placement finds the optimum have only one server location selected in the optimal placement. In this case, the M-GREEDY algorithm outputs the same placement as 1-center. Since the 1-center placement is optimal in only a small portion of all simulation runs, this further suggests the necessity of distributed server placement to achieve better interactivity for continuous DIAs. Similar performance trends are also observed in the results for other numbers of candidate server locations (Figures 10(b) and 10(c)). As the number of candidate server locations increases, it becomes harder to hit a good solution due to larger search space. Figure 10 also shows that the improvement of our M-GREEDY algorithm is even greater in the worst case scenarios. For example, when there are 300 candidate server locations (Figure 10(b)), the 95th percentile results of the M-GREEDY, $k$-center, $k$-favourable and NEAREST server placements are 1.42, 1.86, 3.86 and 6.32 respectively.

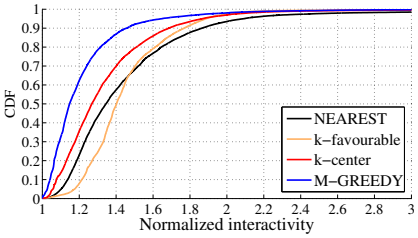
Following similar experimental setup to that in [13], we also evaluate the impacts of server capacity and dynamic network latency on different algorithms in Sections VII-B and VII-C.

### B. Impact of Server Capacity

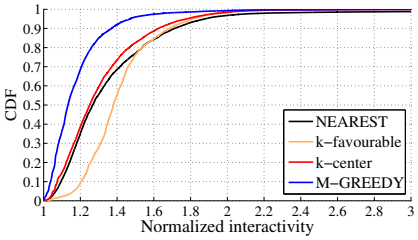
So far, we have not assumed any capacity limitation of the servers. Now, we study the impact of server capacity. Suppose that there are a certain number of servers to be allocated to the locations selected by the algorithms. Each server has a capacity limit on the number of clients that can connect to it. Then, each selected server location can be allocated the number of servers proportional to the number of clients having it as their nearest server locations (subject to integer rounding). The capacity of each location is given by the total capacity of the servers allocated to it. If the nearest server location of a client has been filled to its capacity, the client has to turn to the next nearest server location with spare capacity.

First, we assume that the total server capacity is 900, which is just enough to accommodate the 896 clients in our simulation. Figure 11 shows the performance results for different server capacities and numbers for the simulation runs of 300 candidate server locations. We test 36, 18, 12 and 9 servers each with the capacity of 25, 50, 75 and 100 respectively. It can be seen that the M-GREEDY and M-BETTER algorithms consistently outperform the other algorithms for all the cases tested. We can observe from Figure 11 that the performance of each algorithm slightly deteriorates with the increase of individual server capacity. This is because the actual number of locations with servers deployed cannot exceed the number of servers available. When the individual server capacity increases, the number of servers needed to support all the clients decreases, which implicitly limits the number of server locations used.

Figure 12 shows the impact of the total available server capacity. In this experiment, the capacity of each server is fixed at 50 and we vary the number of servers available for allocation. It can be seen that the M-GREEDY and M-




(a) The connected servers of clients are fixed.




(b) The connected servers of clients change along with the variation of network latency.

Fig. 13. Cumulative distribution of normalized interactivity by different algorithms under dynamic network latency

BETTER algorithms achieve better interactivity performance than the other algorithms. In general, for all the algorithms, the interactivity improves when the number of servers available increases. This is because a larger number of servers increase the capacity of each server location and allows more clients to connect to their preferred nearest server locations. When there are ample capacities allocated to every selected server location, the performance of each algorithm converges to that when there is no capacity limitation at the server locations.

### C. Impact of Dynamics in Network Conditions

The network latency in the real Internet may fluctuate due to the mutability of network conditions. Finally, we study the effect of network latency variation on the performance of different server placements. Since the Meridian dataset does not contain the measurements of network latencies at different times, we use the PlanetLab All Pairs Pings dataset [23] in this experiment. To our knowledge, this is the largest publicly available dataset recording the dynamics of Internet latencies. We have performed simulation runs with the PlanetLab datasets of different days and observed similar performance trends. We present here the results for a sample dataset of 4 June 2005. This dataset contains 96 pairwise latency measurements among 195 PlanetLab nodes at intervals of 15 minutes over a one-day time span.

In each simulation run, we randomly choose 80 nodes as the candidate server locations, and assume that a client is placed at each of the remaining 115 nodes. Each server placement is computed based on the network latencies recorded in the first measurement, and remains unchanged throughout the simulation run. We use two mechanisms for connections between clients and servers. The first mechanism is to connect each client to its nearest server as indicated by the network latencies recorded in the first measurement and keep the connection unchanged regardless of the variation of network latency later. The other mechanism is to ensure that each client always connects to its nearest server based on the latest latency measurement, i.e., the connected server of each client changes with the variation of network latency. The theoretical lower bound on the maximum interaction path length is recalculated at each latency measurement for computing the normalized interactivity. We perform 1000 different simulation runs and plot the cumulative distribution of the normalized interactivity at the times of the 96 latency measurements in Figure 13. Since the M-GREEDY server placement outperforms the NEAREST server placement in most of the simulation runs, the performance of the M-BETTER server placement is close to that of M-GREEDY. For ease of reading, the curve of M-BETTER is not shown in the figure. It can be seen that the M-GREEDY server placement achieves much better interactivity than the other three server placements, for both mechanisms of client connections. Figure 13 also indicates that the M-GREEDY server placement is more resilient to the variation of network latency. For example, if the client connections are fixed (Figure 13(a)), the M-GREEDY server placement achieves a normalized interactivity within 1.5 in 92% of the time. In contrast, the $k$-center, $k$-favourable and NEAREST server placements achieve it in only 79%, 69% and 68% of the time respectively. A similar trend can also be observed when the client connections change with the network latency (Figure 13(b)).

By comparing Figures 13(a) and 13(b), it can be seen that the interactivity performance of each server placement can be slightly improved by adjusting the connections from clients to servers according to the change of network latency. However, the improvement achieved by carefully choosing the locations of servers like M-GREEDY is much more significant. This emphasizes the importance of server placement in improving the interactivity performance of continuous DIAs.

## VIII. RELATED WORK

The classical $k$-center and $k$-median problems have been strongly advocated for server placement in the Internet [14], [24]–[27]. These two problems aim to place $k$ servers in the network to minimize the maximum network latency and the total network latency from the clients to their nearest servers respectively. They well suit the need of web content delivery whose performance is primarily determined by how fast the contents stored on the servers are delivered to the clients. Both the $k$-center and $k$-median problems are NP-hard [20], and have been investigated extensively [27]–[29].

Nevertheless, the $k$-center and $k$-median placements have their limitations in supporting continuous DIAs that are distinguished by the feature of real-time mutual interactions between clients. First, the $k$-center and $k$-median placements do not consider the inter-server latencies, which constitute a non-negligible part of the network latency involved in the interactions between clients. Each client in the DIA connects to only one server but its operations need to be transmitted to other servers for execution through its connected server. Although the latencies from the clients to their servers can be optimized by the $k$-center and $k$-median placements, the

latencies between the servers may be increased at the same time to offset or even outweigh the benefits of the former. Second, neither the $k$-center nor the $k$-median placement gives any consideration to maintaining consistency and fairness for continuous DIAs. To fulfill the consistency and fairness requirements, it is necessary to account for the combined effect of client-to-server latency and inter-server latency [14], [16]. Zhang et al. [16], [30] studied how to optimize the assignment of clients to servers for DIAs given a set of servers placed. Our work here focuses on the orthogonal issue of where to place servers in the network. It will be worth exploring the synergy of these two tuning knobs in the future.

## IX. Conclusion

In this paper, we have investigated the server provisioning problem for optimizing the interactivity performance of continuous DIAs with joint considerations of their consistency and fairness requirements. We have shown that this is a challenging problem by analyzing its hardness under various conditions. We have proved that the problem cannot be approximated within any constant factor for networks without the triangle inequality; within a factor of $3/2$ if the choices of server locations are restricted; and within a factor of $4/3$ if there is a limit on the number of server locations to select. We have proposed two server placement algorithms M-Greedy and M-Better with approximation ratios of 2 and $\frac{5}{3}$ respectively. Experiments with real Internet latency data show that both algorithms produce near-optimal server placements.

## Acknowledgements

This work is supported by Singapore Ministry of Education Academic Research Fund Tier 2 under Grant MOE2013-T2-2-067.

## References

[1] D. Bauer, S. Rooney and P. Scotton. Network Infrastructure for Massively Distributed Games. In *Proc. ACM NetGames*, pages 36–43, 2002.

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

[3] 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.

[4] 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.

[5] 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.

[6] M. Mauve, J. Vogel, V. Hilt and W. Effelsberg. Local-Lag and Timewarp: Providing Consistency for Replicated Continuous Applications. *IEEE Transactions on Multimedia*, 6(1):47–57, 2004.

[7] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot and P. Rodriguez. Greening the Internet with Nano Data Centers. In *Proc. ACM CoNEXT*, pages 37–48, 2009.

[8] S. Choy, B. Wong, G. Simon and C. Rosenberg. The Brewing Storm in Cloud Gaming: A Measurement Study on Cloud to End-user Latency. In *Proc. ACM NetGames*, pages 1–6, 2012.

[9] D. Delaney, T. Ward and S. McLoone. On Consistency and Network Latency in Distributed Interactive Applications: A survey-Part I. *Presence: Teleoperators and Virtual Environments*, 15(2):218–234, 2006.

[10] J. Brun, F. Safaei and P. Boustead. Managing Latency and Fairness in Networked Games. *Communications of the ACM*, 49(11):46–51, 2006.

[11] Y.-J. Lin, K. Guo and S. Paul. Sync-MS: Synchronized Messaging Service for Real-Time Multi-Player Distributed Games. In *Proc. IEEE ICNP*, pages 155–164, 2002.

[12] H. Zheng and X. Tang. On Server Provisioning for Distributed Interactive Applications. In *Proc. IEEE ICDCS*, pages 500–509, 2013.

[13] H. Zheng and X. Tang. Analysis of Server Provisioning for Distributed Interactive Applications. *IEEE Transactions on Computers*, To appear.

[14] 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.

[15] E. Cronin, A. R. Kurc, B. Filstrup and S. Jamin. An Efficient Synchronization Mechanism for Mirrored Game Architectures. *Multimedia Tools and Applications*, 23(1):7–30, 2004.

[16] L. Zhang and X. Tang. The Client Assignment Problem for Continuous Distributed Interactive Applications: Analysis, Algorithms, and Evaluation. *IEEE Transactions on Parallel and Distributed Systems*, 25(3):785–795, 2014.

[17] L. Gautier, C. Diot and J. Kurose. End-to-End Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet. In *Proc. IEEE INFOCOM*, pages 1470–1479, 1999.

[18] 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.

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

[20] 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.

[21] 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.

[22] The Meridian Latency Data Set. http://www.cs.cornell.edu/People/egs/meridian/, 2013.

[23] The PlanetLab All Pairs Pings Data Set. http://pdos.csail.mit.edu/~strib/projects.html, 2013.

[24] 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.

[25] M. Kwok. *Performance Analysis of Distributed Virtual Environments*. PhD thesis, University of Waterloo, 2006.

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

**Hanying Zheng** received the BSc degree in computer science and technology from Sun Yat-Sen University, China, and the PhD degree in computer science from Nanyang Technological University, Singapore. His research interests include computer and communication networks, distributed systems, mobile computing and cloud computing.

**Xueyan Tang** is currently an associate professor in the School of Computer Engineering at Nanyang Technological University, Singapore. He has served as an associate editor of IEEE Transactions on Parallel and Distributed Systems. His research interests include distributed systems, mobile and pervasive computing, and wireless sensor networks. He is a senior member of the IEEE.