Brief Announcement: On Minimum Interaction Time for Continuous Distributed Interactive Computing

Lu Zhang
zh0007lu@ntu.edu.sg

Xueyan Tang
asxytang@ntu.edu.sg

Bingsheng He
bshe@ntu.edu.sg

School of Computer Engineering
Nanyang Techonological University
Singapore 639798

ABSTRACT

In this paper, we study the interaction times of continuous distributed interactive computing in which the application states change due to not only user-initiated operations but also time passing. We formulate the Minimum Interaction Time problem as a combinatorial problem of how the clients are assigned to the servers and the simulation time settings of the servers. We also outline two approaches to approximate the problem.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; G.1.6 [**Numerical Analysis**]: Optimization—*Constrained optimization*

Keywords

Distributed interactive computing; interactivity; consistency

1. INTRODUCTION

Recent years have witnessed rapid development of distributed interactive computing in many areas. In large-scale distributed interactive computing, the application state (such as the virtual worlds in multiplayer online games) is typically maintained across a group of geographically distributed servers [5]. Each participant, known as a client, is assigned to one server and connects to it for sending user-initiated operations. When the application state changes, state updates are delivered to the clients by their assigned servers to reflect the changes. In this way, Distributed Interactive Applications (DIAs) enable participants at different locations to interact with each other in real time.

Interactivity is of crucial importance to DIAs for supporting graceful interactions among participants. The interactivity performance can be characterized by the duration from the time when a client issues an operation to the time when the effect of the operation is presented to others [3]. This duration is known as the *interaction time* between clients. Since the clients interact with one another through their assigned servers, the interaction time between any pair of clients must include not only the network latencies between the clients and their assigned servers, but also the network latency between their assigned servers. These latencies are directly affected by how the clients are assigned to the servers [7]. In addition to network latencies, the interaction time is also influenced by the need for consistency maintenance in DIAs. Consistency means that shared common views of the application state must be created among all clients and it is a fundamental requirement for supporting meaningful interactions [1].

In this paper, we study the interaction times of DIAs. We focus on continuous DIAs in which the application state changes due to not only user-initiated operations but also time passing. In continuous DIAs, the progress of the application state is normally measured along a synthetic timescale known as the *simulation time* (for example, the time elapsed in the virtual game world). To ensure consistency among the application states at the servers, each user operation must be executed by all servers at the same simulation time [4]. As a result, maintaining consistency in continuous DIAs often entails artificial synchronization delays in the interactions among clients. The amount of synchronization delays is dependent on the simulation time settings of the clients and servers.

We formulate the Minimum Interaction Time (MIT) problem for continuous DIAs as a combinatorial optimization problem that includes two sets of variables: the client assignment and the simulation time offsets among servers. We also outline two approaches to approximate the MIT problem: by fixing the client assignment and by fixing the simulation time offsets among servers. In an earlier work [6], we studied some client assignment heuristics for continuous DIAs under a special case of operation execution. This paper generalizes the study by conducting in-depth theoretical analysis of consistency-constrained simulation time settings and achievable interactivity.

2. PROBLEM FORMULATION

A DIA can be modeled by a network consisting of a set of nodes V. A distance $d(u, v)$ is associated with each pair of nodes $(u, v) \in V \times V$, representing the network latency of the routing path between nodes u and v. Denote by $S \subseteq V$ the set of servers and $C \subseteq V$ the set of clients in the network. Each client is assigned to a server for sending user

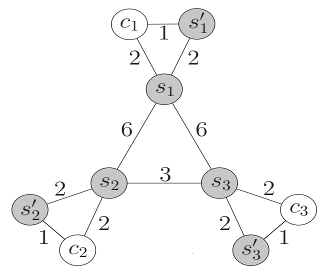


Figure 1: An example network in a DIA.

operations and receiving state updates. For each client $c \in C$, we denote by $s_A(c) \in S$ the server that c is assigned to in a client assignment A.

Each server and client has an associated simulation time to characterize its view of the application state. To provide realistic real-time interaction experiences, the simulation times of all the servers and clients should advance at the same rate as that of the wall-clock time, but they do not have to be synchronized. For each client $c \in C$, we denote by $\delta_c \in \mathbb{R}$ the offset of c's simulation time relative to the wall-clock time (a positive offset means that c's simulation time is ahead of the wall-clock time). Similarly, for each server $s \in S$, we denote by $\delta_s \in \mathbb{R}$ the offset of s's simulation time relative to the wall-clock time.

When a client issues an operation, the effect of the operation is presented to other clients through the following process. First, the client sends the operation to its assigned server. Then, the server forwards the operation to all the other servers. On receiving the operation, each server executes the operation, possibly after some synchronization delay, to compute the new state of the application. Finally, each server delivers the resultant state update to all the clients assigned to it. Since clients inherit the application state from their assigned servers, in order for all clients to always see identical states at the same simulation time, the application states at all the servers must be identical at any simulation time. This in turn requires each user operation to be executed by all servers at the same simulation time, since the state of a continuous DIA changes due to both user operations and time passing. Given a client assignment A and the simulation time offsets of servers $\Delta = \{\delta_s \mid s \in S\}$, our analysis shows that the lowest achievable average interaction time between all pairs of clients that satisfies the above consistency constraint is $D(A, \Delta) =$

$$\frac{1}{|C|}\Big(2\cdot\sum_{i=1}^{|C|} d(c_i, s_A(c_i)) + \sum_{i=1}^{|C|} \max_{s\in S}\left\{d(s_A(c_i), s) + \delta_s\right\} - \sum_{i=1}^{|C|} \delta_{s_A(c_i)}\Big),$$

where $|C|$ is the number of clients. Therefore, we define the Minimum Interaction Time (MIT) problem as follows:

Definition 1. Given a set of servers S and a set of clients C in a network, and the distance $d(u, v)$ between each pair of nodes $u, v \in C \cup S$, the MIT problem is to find a client assignment A and the simulation time offsets of servers Δ that minimize the average interaction time, i.e., to find

$$\min_{A,\Delta} D(A, \Delta).$$

We present an example to illustrate how A and Δ affect the average interaction time. In the network shown in Figure 1, there are three clients c_1, c_2, c_3 and six servers $s_1, s_2, s_3, s'_1, s'_2, s'_3$. A natural configuration is to assign each client c_i to server s'_i (i.e., the nearest server), and synchronize the simulation times of the assigned servers of all the clients, as shown in Figure 2(a), where each client and server is marked with its simulation time offset.[1] Note that the simulation time of each client must lag behind the simulation time of its server due to the network latency of delivering state updates. Suppose that client c_1 issues an operation at simulation time t. As shown in Figure 2(b), the operation first reaches server s'_1 at simulation time $t + 2$, and is then delivered to the other two servers at simulation time $t + 12$. Thus, the operation can be executed by all the three servers at the same simulation time $t+12$ at the earliest, and finally, all the clients receive and present the resultant state updates at simulation time $t + 12$. Therefore, the interaction time from client c_1 to all the clients is 12. Figure 2(c) shows that the interaction time from client c_2 (or c_3) to all the clients is also 12. Consequently, the average interaction time under this configuration is 12.

We can improve the interactivity by tuning the simulation time offsets Δ. Figure 3(a) shows a simulation time setting that reduces the average interaction time to 11. Alternatively, we can also improve the interactivity by tuning the client assignment A. Figure 3(b) shows a client assignment that leads to the average interaction time of 10. The optimal solution to the MIT problem is to tune both A and Δ together as shown in Figure 3(c), which gives the best achievable average interaction time of 9.

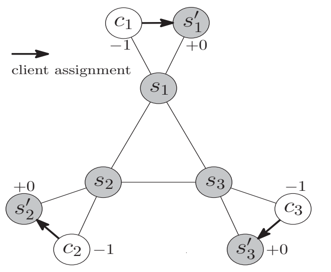
3. RESULTS

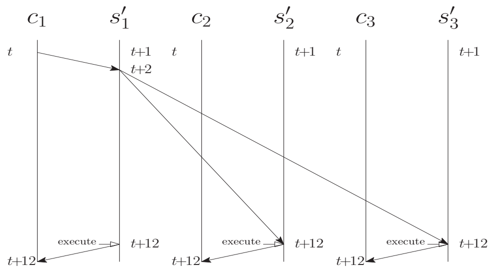
As seen above, the MIT problem is a combinatorial optimization problem with two sets of variables: the client assignment and the simulation time offsets of servers. We can show that the MIT problem is NP-complete. We consider approximating the MIT problem by fixing the client assignment and/or the simulation time offsets. An intuitive and easy-to-implement strategy for client assignment is to assign each client to its nearest server, i.e., the server with the shortest network latency to it. This is known as the *nearest-server assignment* and is widely used in many applications [2]. On the other hand, a simple and straightforward setting of simulation times is to synchronize the simulation times of the servers. Denote such simulation time setting by Δ_0 and denote the nearest-server assignment by N. If N and Δ_0 are employed together, the resultant interaction time $D(N, \Delta_0)$ can be arbitrarily worse than the minimum interaction time $\min_{A,\Delta} D(A, \Delta)$. Interestingly, however, constant approximation factors can be achieved by *either* fixing the client assignment at N *or* fixing the simulation time offsets at Δ_0.

(1) Approximating MIT Problem by Fixing Client Assignment. When the client assignment is fixed, we have the following results about finding the simulation time offsets of servers that minimize the interaction time, i.e., finding $\min_\Delta D(A, \Delta)$.

Definition 2. A *perfect selection* from a $n \times n$ matrix is to select n elements from the matrix, so that there is exactly one element selected in each row and in each column. A

[1]The simulation times of the servers not assigned any client are not really restricted by the consistency constraint. They can be set to lag behind the wall clock time by an arbitrarily large amount and are not marked in the figure.


(a) Natural client assignment and simulation time setting


(b) Interaction time from client c_1

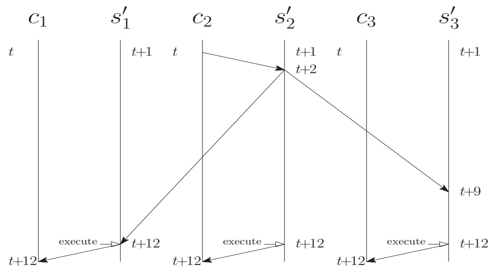
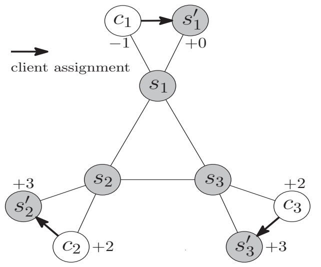
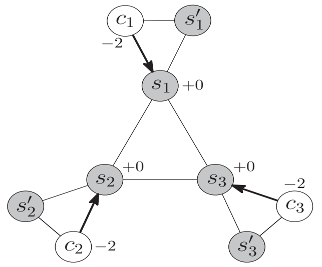

(c) Interaction time from client c_2 (or c_3)

Figure 2: A natural configuration.


(a) Tuning simulation times


(b) Tuning client assignment

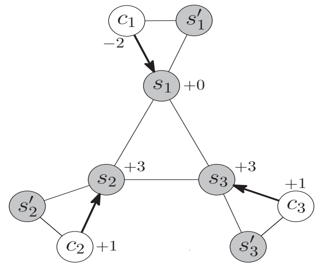

(c) Optimal configuration

Figure 3: Improved configurations.

maximum perfect selection is a perfect selection that has the largest sum of the selected elements among all perfect selections from the matrix.

Definition 3. Given a matrix $\mathbf{Q}$, define $\mathbb{S}(\mathbf{Q})$ as the sum of the elements in a maximum perfect selection from $\mathbf{Q}$.

THEOREM 1. *Given a client assignment A, let $\mathbf{Q}_A$ be a $|C|{\times}|C|$ matrix of $d(s_A(c_i), s_A(c_j))$ $(i, j = 1, 2, \cdots, |C|)$. Then,*

$$\min_{\Delta} D(A, \Delta) = \frac{1}{|C|}\Big(2 \cdot \sum_{i=1}^{|C|} d(c_i, s_A(c_i)) + \mathbb{S}(\mathbf{Q}_A)\Big).$$

We have developed a $O(|C|^2|S|)$ algorithm to compute $\min_{\Delta} D(A, \Delta)$, where $|C|$ is the number of clients and $|S|$ is the number of servers. For networks with the triangle inequality, it can be shown that the minimum achievable interaction time under the nearest-server assignment is within 3 times of the optimal solution to the MIT problem.

THEOREM 2. $\min_{\Delta} D(N, \Delta) \leq 3 \cdot \min_{A,\Delta} D(A, \Delta)$.

(2) Approximating MIT Problem by Fixing Simulation Time Offsets. Finding $\min_A D(A, \Delta)$ is also an NP-complete problem. For networks with the triangle inequality, our analysis shows that this approach can approximate the MIT problem within a factor of 2 if the simulation times of all servers are synchronized.

THEOREM 3. $\min_A D(A, \Delta_0) \leq 2 \cdot \min_{A,\Delta} D(A, \Delta)$.

4. ACKNOWLEDGMENTS

This work is supported by an Inter-disciplinary Strategic Competitive Fund of Nanyang Technological University 2011 for "C3: Cloud-Assisted Green Computing at NTU Campus".

5. REFERENCES

[1] D. Delaney, T. Ward, and S. McLoone. On consistency and network latency in distributed interactive applications: A survey-Part I. *Presence: Teleoperators & Virtual Environments*, 15(2):218–234, 2006.

[2] C. Ding, Y. Chen, T. Xu, and X. Fu. CloudGPS: A scalable and ISP-friendly server selection scheme in cloud computing environments. In *Proceedings of IEEE/ACM IWQoS 2012*, 2012.

[3] C. Jay, M. Glencross, and R. Hubbold. Modeling the effects of delayed haptic and visual feedback in a collaborative virtual environment. *ACM Transactions on Computer-Human Interaction*, 14(2), 2007.

[4] 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.

[5] F. Safaei, P. Boustead, C. Nguyen, J. Brun, and M. Dowlatshahi. Latency-driven distribution: infrastructure needs of participatory entertainment applications. *IEEE Communications Magazine*, 43(5):106–112, 2005.

[6] L. Zhang and X. Tang. The client assignment problem for continuous distributed interactive applications. In *Proceedings of IEEE ICDCS 2011*, pages 203–214, 2011.

[7] L. Zhang and X. Tang. Optimizing client assignment for enhancing interactivity in distribute interactive applications. *IEEE/ACM Transactions on Networking*, 20(6):1707–1720, 2012.