An Effective and Continuous Client to Server Assignment in a Distributed Interactive Application

Rashmi D G
M.tech(CSE)
T.John institute of technology
Bangalore, India.

Mrs. Anju Abraham
Asst.professor, Dept.of CSE
T.John institute of technology
Bangalore, India.

***Abstract*—In distributed interactive applications (DIAs), the interaction time between any pair of clients must include the network latencies between the clients and their assigned servers, and the network latency between their assigned servers as interactivity is a primary performance measure in DIAs. The latencies in the network are directly affected by how the clients are assigned to the servers. The interaction time is also influenced by the consistency and fairness requirements of DIAs. The problem of effectively assigning clients to servers for maximizing the interactivity of DIAs is investigated. The focus is on the continuous DIAs that changes their status not only in response to user operations but also due to the passing of time. Client will search for the nearest server and send request to it. Connection to the server is based on the available memory, resources and work load on the server. To increase the QOS client- monitor stores the popular resources in it, so that the response time to the client is reduced to a great extends and load on the server is reduced. The monitoring system has the available resources in the server and so can help the client if the client requested resource is not available on the connected server. The monitoring system calculates the path length between requested client and server. Newly activated server interact with the other servers and gather information about the nearest clients and clients get connected with the new nearest server. The results show that our proposed Greedy and Modify Assignment algorithms generally produce near optimal interactivity and significantly reduce the interaction time between clients compared to the intuitive algorithm that assigns each client to its nearest server as in existing system.**

Keywords— Distributed interactive application, client assignment, interactivity, consistency, fairness, NP-complete.

I. INTRODUCTION

Distributed interactive applications (DIAs), such as multiplayer online games and distributed interactive simulations, allow participants at different locations to interact with one another through networks. Thus, the interactivity of DIAs is important for participants to have enjoyable interaction experiences. Normally, interactivity is characterized by the duration from the time when a participant issues an operation to the time when the effect of the operation is presented to the same participant or other participants [8]. This duration is referred as the interaction time between participants. Latency in the Network is known as a major barrier to provide good interactivity in DIAs [6]. It cannot be eliminated from the interactions among participants and has a lower theoretical limit imposed by the speed of light. Increasing geographical spreads of participants in large scale DIAs are making distributed server deployment vital for combating the network latency. Latency-driven distribution of servers is essential even when there are no limitations on the availability of server resources at one location .The state of a DIA is often replicated across a group of geographically distributed servers in a distributed server architecture. As shown in Fig. 1, each participant, known as a client, is assigned to one server and connects to the server for sending user-initiated operations and receiving updates of the application state. When issuing an operation, a client first sends the operation to its assigned server. Then, the server forward the operation to all the other servers. On receiving the operation, each server calculates the new state of the application and sends a state update to all the clients assigned to it. Thus, the clients interact with one another through their assigned servers. The interaction time between any pair of clients must include the network latencies between the clients and their assigned servers, and the network latency between their assigned servers. These network latencies are directly affected by how the clients are assigned to the servers. In addition, the interaction time is also influenced by the consistency and fairness requirements of DIAs. Consistency means that shared common views of the application state must be created among all clients to support meaningful interactions [6]. Fairness, on the other hand, is to ensure that all clients have the same chance of participation regardless of their network conditions [8], [3]. Maintaining consistency and fairness in DIAs usually introduces artificial synchronization delays in the interactions among clients due to diverse network latencies [8], [4], [9]. These synchronization delays are also dependent on the assignment of clients to servers. Therefore, how to assign the clients to the servers in DIAs is of crucial importance to their interactivity performance.

In previous work each participant, known as a client, is assigned to one server and connects to the server for sending user-initiated operations and receiving updates of the application state. When issuing an operation, a client first sends the operation to its assigned server. Then, the server forward the operation to all the other servers. On receiving the operation, each server calculates the new state of the application and sends a state update to all the clients assigned to it.

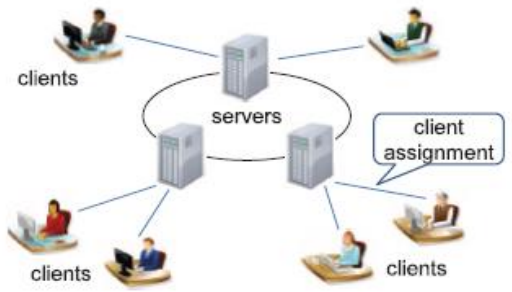

Fig.1. Distributed server architecture

Thus, the clients interact with one another through their assigned servers. The interaction time between any pair of clients must include the network latencies between the clients and their assigned servers, and the network latency between their assigned servers. These network latencies are directly affected by how the clients are assigned to the servers. In addition, the interaction time is also influenced by the consistency and fairness requirements of DIAs. Maintaining consistency and fairness in DIAs usually introduces artificial synchronization delays in the interactions among clients due to diverse network latencies.

a. While Nearest-Server Assignment reduces the client-to-server latencies, it could significantly increase the latencies between the assigned servers of different clients, and thus make the interactivity far worse than optimum.

b. The assumption of the triangle inequality is commonly made when theoretically analyzing the performance of the approximation algorithms in facility location problems.

c. In the absence of the triangle inequality, Nearest-Server Assignment cannot achieve any bounded approximation ratio.

The problem of effectively assigning clients to servers for maximizing the interactivity of DIAs is investigated. Examples of continuous DIAs include distributed virtual environments, distributed interactive simulations, and multiplayer online games. The process start by mathematically modeling the interactivity performance of continuous DIAs under the consistency and fairness requirements. Given any client assignment, the minimum achievable interaction time for DIAs to preserve consistency and provide fairness among clients is analyzed. Based on the analysis, we formulate the client assignment problem as a combinational optimization problem and prove that it is NP-complete. The performance of the algorithms is also experimentally evaluated using real Internet latency data. The results show that our proposed Greedy Assignment and Distributed-Modify Assignment algorithms generally produce near optimal interactivity and significantly reduce the interaction time between clients compared to the intuitive Nearest-Server Assignment algorithm that assigns each client to its nearest server. Distributed-Modify Assignment also has good adaptively to dynamics in client participation and network latency.

II. SYSTEM MODULES

A. Server

The server is connected in order to create a connection between the client and the server where the nodes select there respective server in accordance with the nearest one. The server checks the client which are ideal. Here the client assignment is done using the greedy assignment algorithm. The different servers are interconnected and the servers also perform a checking algorithm where it checks the client which are ideal and also initiates the node having the highest path length to search for the nearest server after all the nodes are connected respectively.

B. Client

Here the clients are deployed in the network and is connected to the server based on the Greedy assignment algorithm. After all the nodes are connected to their respective server the nodes having the highest path length gets a request from their server to update their search so that if any new server is allocated near them then they can connect to that server.

C. Monitor node

The monitor nodes does all the calculation about the nodes pathlength with respective to their nearest server. This calculation gets initiated whenever the server sends a request to the monitor node for updation. After calculation the monitor nodes updates the table of the Server who has sent the request and then with that information the server performs its next task. When all the client connected to the server distributed modification of the client take place .The client with the maximum pathlength is selected and request is sent to that client by the server with which it is connected. After receiving request client will search for the next nearest server for connection. if no new server is found than the client request back to that server where server will search for the idle client. Selected client will take service via idle client from server.

III. RELATED WORK

In paper [1] the author discussed the problem of latency in the network and uses King tool. This tool that accurately and quickly estimates the latency between arbitrary end hosts by using recursive DNS queries in a novel way. It does not require the deployment of additional infrastructure.

In paper [2] the author discussed about the mirror placement problem as a case of constrained mirror placement where mirrors can only be placed on a preselected set of candidates. Performance improvement in terms of client round-trip time (RTT) and server load when clients are clustered by the autonomous systems (AS) in which they reside. the number of mirror sites (under the constraint) effective in reducing client to server RTT and server load.

In paper [3] the propose is Game-independent, network-based service, called Sync-MS, that balances the trade-off between response time and fairness. Sync-MS uses two mechanisms: Sync-out mechanism properly queue up the message at player stations and deliver it to the game application only after the same update message has arrived at

all player stations**.** Sync-in mechanism enforce a sufficient waiting period on each action message dynamically in order to guarantee fair processing of all action messages.The fairness requirement is to ensure that all clients have same chance of participation regardless of their network conditions.

In paper [4] Existing online multiplayer games typically use a client-server model, which introduces added latency as well as a single bottleneck and single point of failure to the game. Distributed multiplayer games minimize latency and remove the bottleneck, but require special synchronization mechanisms to provide a consistent game for all players. A new synchronization mechanism, trailing state synchronization (TSS), which is designed around the requirements of distributed first-person shooter games. Trailing state synchronization (TSS) is designed to execute commands quickly while at the same time maintaining a consistent copy of the game state at all players. When inconsistency does occur due to jitter, the application state can be repair by trailing state synchronization.

In paper [5] the drawback is a novel distributed algorithm that dynamically selects game servers for a group of game clients participating in large scale interactive online games. The goal of server selection is to minimize server resource usage while satisfying the real-time delay constraint. Develop a synchronization delay model for interactive games and formulate the server selection problem. The proposed algorithm, called zoom-in-zoom-out**,** allow the clients select appropriate servers in a distributed manner

In paper [6] the author discussed about Collaborative virtual environments (CVEs) enable two or more people, separated in the real world, to share the same virtual 'space'. CVEs is compromised by one major problem: the delay that exists in the networks linking users together .The 'Impact-Perceive-Adapt' model of user performance, which considers the interaction between performance measures, perception of latency and the breakdown of the perception of immediate causality, is proposed as an explanation for the observed pattern of performance.

IV. PROBLEM STATEMENT

The client assignment problem for maximizing the interactivity of continuous DIAs is formulated as follows: Given a set of servers S and a set of clients C in a network, find a client assignment that minimizes the maximum length of interaction paths between all client pairs, i.e., to minimize

$$D = \max_{c_i, c_j \in C} \{d(c_i, s_A(c_i)) + d(s_A(c_i), s_A(c_j)) + d(s_A(c_j), c_j)\}.$$

V. HARDNESS RESULT

Theorems 1 and 2 below present the hardness results of the client assignment problem.

Theorem 1. The client assignment problem is NP-complete.

Theorem 2. No polynomial-time algorithm for the client assignment problem can achieve an approximation ratio less than 4/3 if the network latency satisfies the triangle inequality and any bounded approximation ratio otherwise, if P ≠NP.

VI. HEURISTIC ALGORITHMS

There are three heuristic algorithms where in the proposed system the combination of greedy and distributed modify assignment algorithm are considered to effectively assign the client to server in order to increase interactivity performance . in proposed system greedy and modify assignment algorithm is used.

A. Dijkstra algorithm

The first algorithm is called Nearest-Server Assignment which intuitively assigns clients to their nearest servers using Dijkstra algorithm. This algorithm can be implemented by having each client measure the network path length between itself and all servers, and select the server with the minimum path length as its assigned server.

After assigning the client to the nearest server the client broadcast the information about its assignment to the neighbor clients that are present within the range.

Algorithm : Dijkstra algorithm

```
 1  function Dijkstra(Graph, source):
 2
 3  dist[source] ← 0                          // Distance from source to source
 4  prev[source] ← undefined                  // Previous node in optimal path
initialization
 5
 6     for each vertex v in Graph:            // Initialization
 7        if v ≠ source                       // Where v has not yet been removed
from Q (unvisited nodes)

 8           dist[v] ← infinity               // Unknown distance function from
source to v
 9           prev[v] ← undefined              // Previous node in optimal path from
source
10        end if
11        add v to Q                         // All nodes initially in Q (unvisited nodes)
12     end for
13
14     while Q is not empty:
15        u ← vertex in Q with min dist[u]   // Source node in first case
16        remove u from Q
17
18        for each neighbor v of u:          // where v has not yet been removed
from Q.
19           alt ← dist[u] + length(u, v)
20           if alt < dist[v]:               // A shorter path to v has been found
21              dist[v] ← alt
22              prev[v] ← u
23           end if
24        end for
25     end while
26
27     return dist[], prev[]
28
29  end function
```

B. Greedy and modify Assignment algorithm

The second algorithm Greedy assignment adopts a greedy approach to assign clients iteratively, starting with an empty assignment. In each step, the algorithm considers all the possibilities of assigning an unassigned client to a server. If a client c is selected to be assigned to a server s, then all unassigned clients that are not farther from s than c are also assigned to s as this would not increase the maximum

interaction path length. Let Δn be the number of new clients assigned to s in this way, and Δl be the increase in the maximum interaction path length due to these new assignments. To minimize the amortized increase in the maximum interaction path length, we use Δl=Δn as the cost metric for selecting which client to be assigned to which server. In each step, among all possible pairs of unassigned client and server, the pair resulting in the minimum cost Δl=Δn is selected and the corresponding clients are then assigned to s. The algorithm terminates when all clients have been assigned to servers. To calculate Δn efficiently, the distances from all clients to each server s are sorted in a list Ls in a preprocessing stage. This sorted list is then incrementally updated by removing newly assigned clients at the end of every step. As a result, Δn can be obtained directly from the index of the unassigned client in the list. On the other hand, Δl for assigning a new client c to a server s is calculated by comparing the maximum interaction path length before assigning c with the maximum length of the interaction paths from c to all the clients already assigned. The latter is given by where 2d(c,s) is the interaction path length from c to itself and C0 is the set of clients already assigned. For each server the term s is independent of client c, so its calculation can be shared among all unassigned clients. The pseudocode of Greedy Assignment is presented in Algorithm 1. Greedy Assignment is suited for centralized implementation due to its need for global knowledge about the distances between clients and servers.

Algorithm 2: Greedy and modify assignment algorithm

1. Initialize C_i=0;
2. for all $C_i \in$ C do
3. pos(C_i) ⟵ info(x, y, C_i);
4. Send Pos(C_i) to monitor node
5. Monitor node calculate NS for C_i
6. Dijkstra (Network graph , Source) ;
7. Monitor node response C_i
8. C_i send request to $S_i \in$ NS
9. if $S_i \neq$ Max (capacity of C)
10. S_i ⟶ response to C_i
11. C_i connect to S_i
12. Check for $N_{c_i} \in C_i$
13. if (check == true)
14. Connect N_{c_i} to S_i
15. i ⟵ i+1;
16. for all j ⟶ 1 to |C| do
17. MaxPL ⟵ MPL(x, y, C_j);
18. If MaxPL ≠ 0
19. posi ⟵ MaxPL
20. while ($S_j \neq 0$)
21. calculate pathlength ⟵ length (C_j, S_j);
22. if pathlength < MaxPL
23. status ⟵ request ($S_j C_j$);
24. if (status == active)
25. assign C_j to S_j
26. else Assign C_j to S_{j+1}
27. else for all k=0 to |C|
28. Initiate monitor node check ⟶ idle client
29. if(check== true)
30. Disconnect C_k from S_j
31. Connect (C_j to S_j);
32. k ⟵ k+1;

The unassigned clients are considered and among that one client will search for nearest server and send requested to that server using algorithm. When server will get the requested it will forward it to the monitoring node for calculation of path length. If the interaction path length is minimum then client get assigned to the server.

Modify Assignment is performed in a distributed manner without requiring the global knowledge of the network at any single server. The assignment is continuously modified for reducing the maximum interaction path length D until it cannot be further reduced. This process is referred to as the assignment modification. One server is selected which as maximum path length with the connected client to perform assignment modification. The client is requested to search next nearest server. If the client finds nearest server then it connect to that server. If two or more servers is selected to perform assignment modifications concurrently, the maximum interaction path length is not guaranteed to reduce because the calculation of each assignment modification is based on the assumption that the assigned servers of other clients remain unchanged.

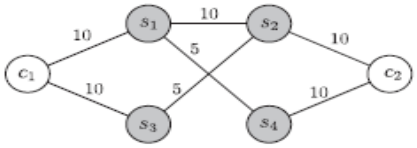

Fig. 3. An example in which changing the assigned servers of two clients simultaneously increases the maximum interaction path length.

The Figure 3 gives an example. Suppose that clients c1 and c2 are initially assigned to servers s1 and s2, respectively, so that the maximum interaction path length is 30. If c1 (or c2) changes its assigned server to s3 (or s4), the maximum interaction path length would be reduced to 25. However, if both clients change their assigned servers, the interaction path length between c1 and c2 would become 40, which is even longer than the maximum interaction path length of the to its clients. Distributed-Modify Assignment has unbounded approximation ratio if it starts with an arbitrary initial assignment, even for networks with the triangle inequality. On the other hand, if Distributed-Modify Assignment takes Nearest-Server Assignment as the initial assignment, the resultant assignment cannot be worse than the latter since the assignment modification can only reduce the maximum interaction path length.

VII. SERVER CAPACITY CONSTRAINTS

So far, the assignment algorithms have not assumed any capacity limitation at the servers. These "uncapacitated" algorithms are suitable for the scenario where each server site has abundant server resources or server resources can be added to these sites as needed. However, if the server capacity at each site is limited, assigning more clients to a server than its capacity may result in significant increase in the processing delay at the server, damaging the interactivity of the DIA. Therefore, we now discuss how to adapt each proposed assignment algorithm to deal with server capacity constraints.

- Greedy Assignment: When selecting the pair of unassigned client and server in each step, the algorithm considers unsaturated servers only. After a client c is selected to be assigned to a server s in a step, if the algorithm cannot assign to server s, all clients closer to s than client c due to the capacity constraint of s, only a portion of these clients are assigned to server s to fill it to capacity. Accordingly, the calculation of Δn is adjusted to reflect the capacity limitations of the servers.
- Modify Assignment: At each assignment modification, a client is allowed to be reassigned to unsaturated servers only.

VIII. CONCLUSION

The client assignment problem for interactivity enhancement in continuous DIAs is investigated. The interactivity performance of continuous DIAs under the consistency and fairness requirements is modeled. The minimum achievable interaction time between clients is analyzed and used as the optimization objective in our formulation of the client assignment problem. The problem is proven to be NP-complete. Three heuristic assignment algorithms are presented. The results show that: 1) our proposed Greedy Assignment and Distributed-Modify Assignment algorithms significantly outperform the intuitive Nearest-Server Assignment algorithm; 2) Distributed-Modify Assignment requires only a small proportion of clients to perform assignment modifications for improving interactivity; and 3) Distributed-Modify Assignment has good adaptivity to dynamics in both client participation and network latency.

To deal with asymmetric routing [6], the network can be modeled by a directed graph. Each pair of nodes is associated with the lengths of two routing paths of different directions. The interaction path from a client ci to another client cj can be considered as a directed path that is different from the interaction path from cj to ci. It is easy to show that if we change the definition of D to be the maximum length of all the directed interaction paths between clients, the consistency and fairness requirements can still be satisfied. Therefore, the objective of the client assignment problem becomes to minimize the maximum length of all directed interaction paths. For the heuristic algorithms, we can simply use the lengths of the directed routing paths between clients and servers in the calculation without modifying the algorithms. However, the approximation ratios of the algorithms may change. In future when the new server is deploy it get the information about the client from other servers. After getting the information it will search for the nearest client based on the information and it will send the request to that client. After receiving the request from the server client will get assign to the newly deployed server.

REFERENCES

[1] K.P. Gummadi, S. Saroiu, and S.D. Gribble, "King: Estimating Latency between Arbitrary Internet End Hosts," Proc. Second ACM SIGCOMM Workshop Internet Measurement, pp. 5-18, 2002.
[2] E. Cronin, S. Jamin, C. Jin, A.R. Kurc, D. Raz, and Y. Shavitt, "Constrained Mirror Placement on the Internet," IEEE J. Selected Areas Comm., vol. 20, no. 7, pp. 1369-1382, Sept. 2002.
[3] Y.J. Lin, K. Guo, and S. Paul, "Sync-MS: Synchronized Messaging Service for Real-Time Multi-Player Distributed Games," Proc. IEEE 10th Int'l Conf. Network Protocols (ICNP '02), 2002.
[4] E. Cronin, A.R. Kurc, B. Filstrup, and S. Jamin, "An Efficient Synchronization Mechanism for Mirrored Game Architectures," Multimedia Tools and Applications, vol. 23, no. 1, pp. 7-30, 2004.
[5] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker, "On Routing Asymmetry in the Internet," Proc. IEEE Global Telecomm. Conf. (GLOBECOM '05), 2005.
[6] 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, vol. 15, no. 2, pp. 218-234, 2006.
[7] J. Brun, F. Safaei, and P. Boustead, "Managing Latency and Fairness in Networked Games," Comm. ACM, vol. 49, no. 11, pp. 46-51, 2006.
[8] C. Jay, M. Glencross, and R. Hubbold, "Modeling the Effects of Delayed Haptic and Visual Feedback in a Collaborative Virtual Environment," ACM Trans. Computer-Human Interaction, vol. 14, no. 2, article 8, 2007.
[9] K.W. Lee, B.J. Ko, and S. Calo, "Adaptive Server Selection for Large Scale Interactive Online Games," Computer Networks, vol. 49, no. 1, pp. 84-102, 2005.
[10] M. Marzolla, S. S. Ferretti, and G. D'Angelo, "Dynamic Resource Provisioning for Cloud-Based Gaming Infrastructures," ACM Computers in Entertainment, to be published.
[11] M.R. Korupolu, C.G. Plaxton, and R. Rajaraman, "Analysis of a Local Search Heuristic for Facility Location Problems," J. Algorithms, vol. 37, no. 1, pp. 146-188, 2000.
[12] C. Lumezanu, R. Baden, N. Spring, and B. Bhattacharjee, "Triangle Inequality and Routing Policy Violations in the Internet," Proc. 10th Int'l Conf. Passive and Active Network Measurement (PAM '09), pp. 45-54, 2009.