

Improving Online Gaming Experience Using Location Awareness and Interaction Details

Dewan Tanvir Ahmed and Shervin Shirmohammadi

Distributed and Collaborative Virtual Environments Research Laboratory

School of Information Technology and Engineering

University of Ottawa, Ontario, Canada

{dahmed, shervin}@discover.uottawa.ca

Abstract — Latency is a key element for online game quality and user experience. The client-server approach is a widely used system supporting hundreds of thousands of players on a regular basis. However, if latency among players via an intermediate server is large, timely interaction for them could be difficult. In this article, we present a procedure to share game states for a group of players within their area of interaction so that players can comply with stringent time-constraint and improve their game experience. As players move around in a game space, so do their virtual positions. In addition, the relative orientation of players within an area of interaction is unpredictable which indeed changes quite frequently. Because of these facts, we cannot make a predefined rule set for message exchange among players. So a message exchange plan currently working well might have low efficiency after a while due to dynamic changes. In our procedure, considering the importance of interaction, relative orientation, and virtual and geographical locations, we devise a message exchange plan that works alone in each client machine with the local information available. Significant performance improvements are noticed through simulations, validating our approach.

Keywords- *Networked Games; client-server; latency; lag compensation; synchronization; performance*

I. INTRODUCTION

Internet-based multiplayer online games, especially First Person Shooter (FPS) games such as Counter-Strike: Source, and Quake III Arena introduced over a decade ago - have become increasingly popular and significant contributors to Internet traffic [1]. Online games usually function in a client-server mode where players control game clients generally running on a personal computer (PC) that communicates with game servers hosting individual games. Because of significant resource requirement, game publishers rely on Internet Service Providers (ISPs), dedicated game hosting companies, and private individuals to host game servers.

Massively Multiplayer Online Games (MMOG) support millions of subscribers and many of them actively participate in games on a daily basis. To manage this large number of players, the virtual world is typically divided into realms or kingdoms which are the clones of the same virtual world, each hosting several thousand registered players. To be precise, realms are geographically distributed across the Internet. So, players from one particular region generally participate in the same realm. Realms are further divided into separate areas.

Each area is considered as a zone. Zones can have different themes and different level of difficulties — in order to hold inexperienced players advancing into the next hard level. Normally, there is a server for each zone managing game traffic. The communication strategy within a zone best resembles to a multicast structure because of players' common interest (i.e. regular interaction) in game logic. IP multicasting can be an ideal solution which was initially proposed for group communication. But IP multicast is designed for hierarchical routing and does not scale well in terms of supporting a large number of concurrent groups. The deployment hurdles and service provider's aversion to provide IP multicast make matters worse [2][3]. Current practices therefore deeply rely on centralized architectures that cause scalability bottlenecks (in terms of maximum number of people supported in a single realm) and are also costly to adopt and deploy. To address this problem, new designs are proposed attempting to include both client and server side resources in a seamless manner to take advantage of the P2P paradigm [2][4], with the objective of utilizing client side resources, quick deployment and better service. In this paper, we assume a client-server infrastructure, as currently this is the most widely-deployed method to support MMOGs.

In order to support a consistent game space, each player must keep a clone of relevant game states in his computer. When a player performs an action or generates an event affecting the virtual space, the update must be shared with other players around. The amount of data required to exchange roughly depends on population size of the interested area. However, the capacity is bounded by at least two practical limitations — network bandwidth and processing power [5][6]. Latency tolerance usually varies from game to game and is typically limited to a value between 100ms to 1000ms depending on many factor such as game perspective (i.e. First-person or Third-person), game genres (i.e. racing or role playing game), and the sensitivity of the actions [7].

In this article, we devise a game state sharing mechanism that helps players to obey time constraint of the application. In online games, virtual and physical positions are two important terms. The first one defines a player's location in the game space while the latter represents the actual position of a player's computer in the Internet (as a node in the network). Both are significant and play a role in our design. We believe that in the same interaction space, the importance of

interaction among players is non-uniform. Simply, a closer player has higher importance than a distant one, which is intuitive in some sense. On the other hand, symmetric and asymmetric relationships (explained later) among players are not fixed in a game space and change over time based on players' relative position and orientation. So a message prioritization procedure should take these temporal symmetric and asymmetric relationships into account while formulating rules to exchange game states. Considering the importance of interaction, relative orientation, and virtual and geographical locations, we propose a message exchange plan that works standalone in each client machine with the local information available.

After giving an overview of related work in the following section, we state the problem and objectives in Section III. In Section IV, we introduce the algorithm to comply with the time constraint. In Section V, a quality control mechanism is outlined. The new message exchange plan for online games is presented in Section VI. The performance analysis and other relevant issues are discussed in Section VII. Finally, the paper is concluded in Section VIII.

II. RELATED WORK

The early proposals of online games - suggested decades ago - were purely Client-Server in nature where an update message is sent from one player to the server who then relays it to all other players. While peer-to-peer (P2P) approaches have recently been proposed to either move away from a client-server approach or to complement it, it is apparent that a pure P2P architecture is not a viable solution by gaming companies due to business prospect, security, and quality concerns. Also, there are serious technical doubts about the practicality of a purely P2P approach for MMOGs [8]. At present, different mixed or hybrid architectures are being proposed using peer resources, but practical deployment hurdles are not yet fully overcome.

Consistency, responsiveness, reliability, security, and persistency are fundamental concerns of MMOGs [9][6]. Real-time applications like online games demand realization of an action in time. The general phenomenon is that, when a player interacts with other players, updated information must be sent to all the participants. Network latency and processing delay make the matter hard. Because of networking limitations and traffic conditions, some of these updates might be lost or delayed. Much research has been conducted to overcome the networking limitations and to make more reliable systems. Some of these studies provide receiver-initiated and selectively-reliable transport protocols [10] that can be used to deliver important messages with a high degree of reliability, while others use sender-initiated approaches to deliver key updates with guaranteed reliability [11]. The IEEE DIS standard [12] has also been successfully used in a controlled environment with vast resources, mostly for military simulations.

Intra-zone communication means communication inside a zone or within a group. The players who are close to each other in a virtual world interact more frequently than others

and are involved in many common activities. In order to retain consistency, there is a need to exchange messages among themselves, and intra-zone communication becomes necessary. For local communication inside a zone, the zone master's and players' active participations are combined to overcome the resource limitations of the master. This is a kind of overlay-based state-sharing mechanism. As mentioned earlier, the acceptable latency value for online games appear to be game specific and defined by upper bounds. For example, Fritsch et al. show that the game *Everquest2* can run with a latency up to 1250ms [13]. For this purpose, authors monitor the time span in which players kill a certain number of monsters, outstanding health, and magic points that players have after the encounter. Dick et al. conduct player survey and analyze how latency affects player performance in different games [14]. According to this, a *Role Playing Game* like *Diablo II* performs well when latency is around 80ms which can be stated as the optimal, and a latency of 120ms as a maximum tolerable value. While the client-server paradigm is easy to deploy and is commonly used, the fact that a message must go through an intermediary node (server) to reach its destination can add unwanted delay. In this article, we present a state sharing procedure that allows players to exchange game state directly when the experienced latency via intermediate server exceeds the threshold.

Marios et al. present an approach to support massively multiplayer online role-playing games (MMORPGs) using a centralized distributed architecture [15]. This approach considers a player's locality of interest to reduce bandwidth requirements for both game servers and clients. But from an architectural point of view, it is simply a multiple server-based client-server architecture where performance improvement is flat. There is no guarantee on end-to-end delay. Here, a player state includes a set of all other players and servers that currently know this player. However, if a player leaves, it is not clear what will happen to others, i.e. how will this departure be handled? This effect of player departure is not addressed in this architecture.

The model proposed by Hampel et al. reuses architectures capable of exploiting the flexibility and scalability of P2P networks [16]. One of the main drawbacks of P2P networks for games is the lack of a central authority that can regulate access and prevent cheating. The model in [14] overcomes that by using a set of controller peers that can supervise each other. This kind of redundancy can prevent cheating. The model is based on the existing distributed hash table Pastry, which has been extended into SCRIBE. The key issue is unbounded end-to-end delay, which could be a problem for synchronization in online games.

Our proposed approach works in client-server mode but includes active participation of players when latency affects gaming experience. Out of many players, important pairs of players are identified considering attributes such as player's geographical placement, virtual position in the game space, and their relative orientation. Earlier we have considered only physical and virtual positions of players while developing the message forwarding mechanism [17]. In this article, we also

consider the importance of the location of interaction and introduce a new game state sharing procedure that will improve gaming experience.

III. RESEARCH OBJECTIVES

The game space of MMOGs contains plenty of information, but a player requires only a small subset of that information. For online games, *Area of Interest Management* (AoIM) is a technique used to reduce communication overhead. AoIM methods intelligently determine useful information for each player and block irrelevant information. For example, the area of interest of an avatar in an MMOG is the set of avatars and *non-playing components* (NPC) with whom it interacts. Since the virtual world of a game is large, filtering out unrelated information is a fundamental requirement for this type of applications. Thus, passing relevant information to the players is an effective way to approach messaging in online games. Let an MMOG server (S) manage an *area of interest* (AOI) of n players where they communicate with each other. Each time a message is sent to the server, it is relayed to all destinations by the server. Let P be a set of players, and l_i be the end-to-end delay between player p_i and the server S . Also, let the set $L = \{l_1, l_2, \dots, l_n\}$ define the end-to-end delay from the server to the players. Thus, the time required to forward a message from a player p_i to another player p_j through the server is $L_{ij} = l_i + l_j$. Considering the processing delay (D_p) at the server, this becomes $L_{ij} = l_i + l_j + D_p$. Thus, the pair-wise latency of all pair of players through a single hop server can be represented by the matrix L_M :

$$\begin{bmatrix} 0 & \dots & L_{1n} \\ \vdots & \ddots & \vdots \\ L_{n1} & \dots & 0 \end{bmatrix} \dots \quad (1)$$

The maximum latency observed can be determined by picking up the two highest latencies to the server. Say, l_{M_1} and l_{M_2} are the two highest latencies to the server for players P_{M_1} and P_{M_2} , respectively. So, the maximum latency will be $L_{max} = L_{M_1 M_2} = l_{M_1} + l_{M_2} + D_p$. The average latency can be defined by equation (2) considering symmetric latency relationship among players,

$$A_L = \frac{\sum_{i=1}^n \sum_{j=i+1}^n l_{ij}}{\frac{n(n-1)}{2}} \dots \quad (2)$$

Let T_C be the latency constraint of the application; i.e., the maximum acceptable delay threshold beyond which the game is considered unplayable. When the latency between two players is close to the latency constraint of the application, the situation can be regarded as a critical stage. Failure to satisfy T_C may adversely affect the performance and degrade gaming experience. So, the performance will remain steady as long as $L_{ij} < T_C$ for all i and j . Our object is to find out answers and possible solutions for the following concerns:

1. What will happen when the observed latency exceeds the application specific threshold i.e. $L_{ij} > T_C$? Is there a way to handle this situation?

2. How can we improve gaming experience when latency is at a critical stage?
3. What are the possible components to consider enhancing user experience? How can we include these into our algorithms?

The solution of the first question can be found in Section IV. Both Sections V and VI cover the answer of the last two questions.

IV. CONSTRAINT AND THE PROPOSED APPROACH

A. Peer Potentials

One of the big problems of group communication is the shortage of peer's bandwidth. In P2P applications, the peers (or participants) share their resources. Generally, a peer can be different from others based on various characteristics such as bandwidth and CPU power. This is called peer heterogeneity making the problem complicated. Sharing of game states through peers is considered in this paper. But peers cannot afford to establish connection to all other peers because of their limited bandwidth. The proposed system requires an estimate of the committed peer's bandwidth. In this architecture, players can exchange state information directly, while in parallel they forward their updates to the server. Thus, the system roughly estimates a number for each player, implying peer message forwarding capabilities. We call this *peer potential* or *degree*. In short, the term 'degree' means the number of players to whom a player can communicate directly according to its available bandwidth. For this purpose, we divide a peer's upload bandwidth by the bandwidth required to serve each player per unit time (BwP) which is an application specific parameter. Thus, degree can be defined as $d = \text{uploadBW} / BwP$. We also define the term *used-degree* as the number of degrees which are currently in use.

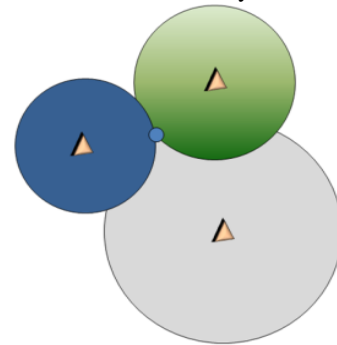


Figure 1. Approximating location through landmarks

B. Geographic Location Approximation

To better comply with game specific time constraints, we need to know each player's physical position. Using the HELLO packet method for latency estimation is a common technique, but it is not scalable for a larger number of players. An approximation concept can be used where each player determines its distance to a few geographical landmarks placed strategically in the system. The distances to the landmarks are going to be used for position estimation. It

should be noted that these landmarks are not necessarily servers. It is intuitive that three landmarks would be adequate for such approximation as shown in Figure 1. In this figure, the triangles are the landmarks while the blue dot represents the node whose location will be approximated. As the end-to-end delay itself is not invariant, the intersections may not be precisely available. Thus, the physical location of players' machine in the global coordinate system is roughly approximated. The key benefit of the geographic location comes from its quicker approximation as we do not need to exchange explicit overhead messages among players to calculate distance or delay [18].

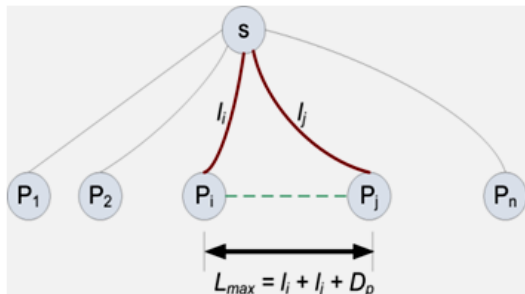


Figure 2. The approach to satisfy time-constraint.

Algorithm *Time-Constraint-Satisfaction*

Input

N : The total number of players

P : The set of players

D_p : Packet processing delay at sever

T_C : Application specific time constraint

begin

for each player $p_i \in P$ **do**

 find geographical location (G_i) using land marks

 define end-to-end delay to the server, l_i

 define degree (d_i)

 set used-degree (u_i) to 0

end

form the latency L_M using $L_{ij} = l_i + l_j + D_p$

determine Pair-Wise-Delay $PWD = \{L_{i,j} | \forall i,j i < j\}$.

sort PWD in descending order

do while ($\text{Max}(PWD) > T_C$)

 mark Player P_i and P_j that introduce $\text{Max}(PWD)$

if ($d_i > u_i$ and $d_j > u_j$) **then**

$u_i = u_i + 1$

$u_j = u_j + 1$

 adjust PWD using G_i and G_j

else 'relaxation is not possible' and exit

end

end

Figure 3. Approaching critical time-constraint

C. Message Redirection Procedure

For a system with n players, there are $C(n, 2)$, i.e. $\binom{n}{2}$, unique pairs among them. Thus, matrix L_M in equation 1 has $C(n, 2)$ non-zero entries representing the latency of pairs. Let Pair-Wise-Delay (PWD) be a data structure (an array or a matrix) that keeps such information. This data structure considers a symmetric latency relationship among players. Let us assume Players p_i and p_j introduce the largest latency $L_{max} = L_{ij} = l_i + l_j + D_p$. When L_{max} crosses the threshold limit i.e. $L_{max} \geq T_C$, then one way to satisfy time-constraint is to allow the involved peers to exchange game states directly (Figure 2). In this figure, S stands for a server, p_i represents a player and l_i means the latency between S and p_i . Thus, player p_i not only forwards update messages to the server but also to player p_j , and vice versa. The server relays this message to other players performing the regular task of interest management. The procedure can be applied to the next highest L_{max} subject to available of resource. The main idea is to allow exchange of game states directly between those players when there is a large pair wise latency, instead of following a comparatively long path through the server. The algorithm given in Figure 3 stops the execution if the current relaxation step¹ is unsuccessful. The algorithm can continue further for other pairs where $\text{Max}(PWD) > T_C$. We are not looking for an optimal solution. This is rather a practical and deployable alternative. Later in Section VI, we present the modified and extended version.

V. QUALITY ENHANCEMENT

A. Virtual Position and its Impact

As mentioned before, in the game, the virtual position refers to a player's location in the virtual world or in the game field, while the physical position means the actual position of the player's node in the Internet. Both are important and should be considered while designing an MMOG system:

- If two players are from two distant physical positions located far away from each other, their interaction in the virtual world could be challenging, and timely sharing of game states could be problematic. Certainly, physical position influences game performance which must be taken into consideration while offering the game service.
- The virtual position of players in the game defines their relative closeness. The area of interest is a well-known term in the gaming community and forms a group for a set of closer players in terms of virtual position. But we believe that even in the same area of interest, the importance of interaction for two players is not uniform. Simply, a closer player has higher importance than a distant one. For a better gaming experience, the importance of interaction must be taken into consideration which generally depends on their virtual position.

We therefore make the following intuitive assumption which holds for most games and situations:

¹ In order to avoid the long latency introduced by an intermediate server, the direct communication of two users with each other is called a relaxation step.

Assumption - In an area of interest, the importance (or value) of interaction between two players is inversely proportional to their virtual distance, i.e. $I_{i,j} \propto \frac{1}{\text{virtual_distance}_{i,j}}$.

So, a player should treat other players differently based on their virtual distance. We state that a virtually closer player will get more importance than a distant player because of their frequent interaction. Considering this principle, a *Closeness matrix* (Cl) is formed for a group of players that keeps such distance information representing their relative virtual distances. So, players in a group keep their closeness information and give importance accordingly while forwarding game states to others.

B. Quality Control Mechanism

Quality control is an integral part of any system. For better gaming experience, the latency and closeness matrices formed earlier can be consulted to identify where the quality improvement will be significant. As two different metrics are considered for quality control, we normalize the latency matrix L_M by T_c (maximum acceptable delay threshold)

$$N_L = L_M / T_c$$

The closeness matrix can be constructed from virtual distances among players as explained earlier. Let $d_{i,j}$ present virtual distance between player p_i and player p_j . So, the distance matrix is

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,n-1} & d_{1,n} \\ d_{2,1} & d_{2,2} & \dots & d_{2,n-1} & d_{2,n} \\ \vdots & & \ddots & & \vdots \\ d_{n,1} & d_{n,2} & \dots & d_{n,n-1} & d_{n,n} \end{bmatrix}$$

Let $maxDistance$ be the maximum virtual distance for this set of players. As a result, after normalization, the closeness matrix becomes

$$Cl = \frac{(maxDistance+1)-D}{maxDistance+1} \quad \dots \quad (3)$$

Virtual distance among five players				
0	45	24	89	81
45	0	14	27	35
24	14	0	54	32
89	27	54	0	18
81	35	32	18	0

The value 0 means mp virtual distance, i.e. the player itself

Closeness relation				
1.00	0.5	0.73	0.01	0.10
0.50	1.00	0.84	0.70	0.61
0.73	0.84	1.00	0.40	0.64
0.01	0.70	0.40	1.00	0.80
0.10	0.61	0.64	0.80	1.00

Maximum virtual distance = 89
The value 1 means players self importance, not used for message redirection

Figure 4. Formation of a closeness matrix - an example

Figure 4 shows how the closeness matrix is formed for a group of players. After normalization, the higher value of both matrices indicates higher importance for quality control in terms of two different scores. Because of limited bandwidth, a player cannot make connection to all players of its interest. The quality precedence matrix (Q_p) estimates a value for each pair

of players considering the latency and closeness. In other words, it defines an order of players meaning which pair to be relaxed first and before others. Now based on latency and closeness (virtual distance in a game), the quality precedence matrix $Q_p = wC \times Cl + wL \times N_L$ will be formed with respect to their relative significances (or weight) wL and wC , respectively, where $wL + wC = 1$. The weights are set by the game provider. Generally, we can put a higher weight on wL if players are geographically far apart. The game engine can tune Q_p according to its requirements. The quality control algorithm is given in Figure 5. The latency is estimated using the geographical location of the players, as explained in Section IV(B). According to this procedure, each pair of players has an estimated precedence or importance. The importance level (I_L) guides us at what level of importance we can allow direct exchange of messages. This could be defined by the game service provider, e.g. $I_L = 0.7$. A higher value has higher significance. As the closeness matrix changes frequently, so does the quality precedence matrix. We suggest a periodic adjustment of the second part of the quality control algorithm.

Algorithm Quality Control Algorithm

Input

N : The total number of players

P : The set of player

D_p : Packet processing delay at sever

T_c : Application specific time constraint

I_L : Importance level

begin

for each player $p_i \in P$ **do**

 find geographical location (G_i) using land marks

 define end-to-end delay to server, l_i

 define degree (d_i)

 define used-degree (u_i) to 0

end

form latency matrix L_M using $L_{ij} = l_i + l_j + D_p$

for each pair of players i and j **do**

if ($L_{i,j} > T_c$)

if ($d_i > u_i$ and $d_j > u_j$) **then**

$u_i = u_i + 1$

$u_j = u_j + 1$

 Adjust L_M using G_i and G_j (message redirection)

else 'relaxation is not possible'

end

for each pair of players i and j **do**

if ($Q_{i,j} \geq I_L$ and $Q_{j,i} \geq I_L$) **then**

if ($d_i > u_i$ and $d_j > u_j$) **then**

$u_i = u_i + 1$

$u_j = u_j + 1$

 Adjust L_M using G_i and G_j (message redirection)

else 'resource not available'

end

end

Figure 5. The quality control algorithm

VI. INTERACTION AND MESSAGE EXCHANGE PROCEDURE

Avatars in online games move constantly, so the relationships among players based on virtual distance change regularly. On the other hand, the physical distance, i.e. end-to-end delay, among players remains roughly constant. Combining these two distance features to prioritize message redirection can be advantageous and can improve gaming experience. In an area of interest, each player knows the virtual position of all other players. In the model proposed earlier, the physical locations are approximated when a player starts a new session. So, both locations are known to users. In the following section, we present a new procedure by which players' gaming experience can be improved by combining these distance features.

Before presenting the details, symmetric and asymmetric relationships need to be defined in terms of message importance. In symmetric relationships, game states have equal importance for two players in both directions; i.e., an update about player A for player B is as important as an update about player B for player A. Usually when two players collaborate face-to-face in a game, symmetrical relationships are observed as shown in Figure 6(a). On the other hand, in asymmetric relationships, game states have different consequences between two players. Here an update about player A for player B is not as important as an update about player B for player A. For example, let's say two players A and B running towards west and player B is behind player A. So player B can see player A, but not vice versa. So the position update of player A is important to player B but the position update of player B is not important to player A as shown in Figure 6(b). This is called asymmetric relationship. According to the above definitions, symmetric and asymmetric relationships among players are not fixed for an online game which changes over time based on players' relative position and orientation. So a message prioritization procedure should take temporal symmetric and asymmetric relationships into account while formulating rules to exchange game states.

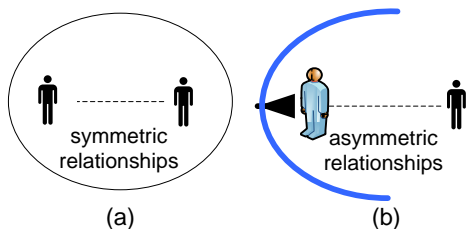


Figure 6. Possible relationships among players (a) Symmetric (b) Asymmetric

Considering the details of earlier discussion and presented problem solutions, we now present a new message sharing procedure that takes the following factors into account:

1. Virtual positions of players within an area of interest
2. Symmetric and asymmetric relationships among players
3. Physical positions of players
4. Players' available resources
5. Time constraint (delay threshold) of the game

The goal of this procedure is to form a message exchange platform so that players can satisfy time constraint and improve gaming experience. Active participation among players within the client-server architecture is considered in this procedure. All five factors can be known and instantiated at runtime. Factors 1 and 2 can be realized through the game states regularly. Factor 3 can be approximated through landmarks when players join. Factors 4 and 5 are application specific parameters which can be set at the beginning as well. The high-level description of this procedure, shown in Table 1, is as follows. Each player independently takes its message forwarding decision considering the importance of inter-player interaction. The details are discussed below:

Precedence 1 The latency between two players A and B through the server exceeds the threshold limit.

Precedence 2 Player A and B are virtually close and non-critically geographically apart but there exists symmetrical relationships.

Precedence 3 Player A and B are virtually close and geographically reasonably apart but there exists a symmetrical relationship between them.

Rules for Precedence 1, 2 and 3

Approach—Player A and B exchange their game states directly

Server tuning – Server will stop relaying updates of player A to player B and vice versa

General Rules

Situation – All cases except the situations of precedence 1-3

Approach –Player A and B exchange game states via game server

Server tuning – Not applicable

Algorithm *Serve-side-Solution-for-Time-Constraint*

Input

n : The total number of players

P : The set of player

D_p : Packet processing delay at sever

T_C : Application specific time constraint

begin

for each player $p_i \in P$ **do**

 approximate geographical (i.e. physical) location (G_i) using landmarks

 define end-to-end delay to server, l_i

 define degree (d_i)

 define used-degree (u_i) to 0

end

form latency matrix L_M using $L_{ij} = l_i + l_j + D_p$

for each pair of players i and j **do**

if ($L_{ij} > T_C$)

if ($d_i > u_i$ and $d_j > u_j$) **then**

$u_i = u_i + 1$

$u_j = u_j + 1$

 adjust L_M using G_i and G_j (message redirection)

else 'relaxation is not possible' for these two players

end

send a message to all players containing the information of geographical position and available resources of the players

end

Figure 7. Server side procedure for time constraint

Table 1. The rules for message exchange procedure

Virtual Distance	Physical Distance	Symmetric	Asymmetric	Communication policy between player A and B
x	critical	x	x	directly
close	large	yes	no	directly
close	large	no	yes	server
large	large	x	x	server
close	average	yes	no	directly
close	average	no	yes	server
large	average	x	x	server
x	close	x	x	server

The aim of this message exchange procedure is to satisfy time constraint and improve gaming experience. In this procedure, players can communicate directly when necessary, considering the importance of their interaction, relative orientation, and virtual and geographical locations. As players move in the game space, so do their virtual positions. Similarly, relative orientation of players within an area of interaction is unpredictable and changes frequently. Because of these facts, we cannot make a unique rule set for message exchange among players. Thus, at any given point in time, the current message exchange plan works well but has less value after a while due to the dynamic changes discussed above.

On the other hand, dependence on the server for a well-defined rule set can be costly due to the frequent refresh requirements. As a solution, the presented method works independently in a distributed manner with the local information available. Having escaped the strict time constraint, the algorithm continues in greedy mode. The server and client side procedures for message exchange dealing with the time constraint and gaming experience are given in Figure 7 and Figure 8, respectively.

Three auxiliary functions (Virtually-Close (C, v_i), Physically-Apart (C, l_i) and Symmetric-Relation (C, o_i)) used in *Client-side-Performance-Improvement* are reasonably simple and computationally inexpensive. The descriptions of these functions are mentioned below. It is important to mention that the algorithms run with the assumption of resource availability; i.e., the algorithm tries the next option if the partners at the other hand has adequate bandwidth or stops immediately if it cannot afford more.

Virtually-Close (C, v_i): This function checks the virtual distance between two players. The return value will be TRUE if they are virtually close based on a predefined threshold value, otherwise it returns FALSE.

Physically-Apart (C, l_i): This function checks the physical distance between two players. The approximated physical positions are determined earlier when players join a game session. It uses two thresholds to differentiate players into three groups. The return values are LARGE, AVERAGE and CLOSE when the physical distance between two players are large, average and small respectively.

Symmetric-Relation (C, o_i): To determine the relationship between players in terms of orientation, this function is used. If there is a symmetric relation between two given players it returns TRUE, otherwise FALSE.

Algorithm Client-side-Performance-Improvement

Input*P*: The set of player in the area of interaction*m*: number of players*G*: Geographical position of players in the area of interaction**Begin**// let *C* stands for a player running this code moduledetermine virtual distance (v_i) from this client to all other playersdetermine physical distance (l_i) from this client to all other players (receive this information from server while physical position approximation)determine orientation (o_i) of players

// Precedence 2

for each player in the area of interaction $p_i \in P$ **do**if (Virtually-Close(C, v_i) AND Physically-Apart(C, l_i)=LARGE AND Symmetric-Relation(C, o_i))Inform the server about their direct collaboration
Player *C* and p_i will exchange game states directly
 $P = P \setminus \{p_i\}$ **end**

// Precedence 3

for each player in the area of interaction $p_i \in P$ **do**if(Virtually-Close(C, v_i) AND Physically-Apart(C, l_i)= AVERAGE AND Symmetric-Relation(C, o_i))Inform the server about their direct collaboration
Player *C* and p_i will exchange game states directly
 $P = P \setminus \{p_i\}$ **end****End**

Figure 8. Client side procedure for time constraint and quality improvement

VII. PERFORMANCE MEASUREMENT AND ANALYSIS

We conducted experiments to verify the proposed concept in the context of online games. For our simulation, we used counter strike data, where the average packet size is 50.4 bytes and 6.15 bytes for each additional client without UDP header

[19][20]. The distribution has been considered to have a heavy tail behavior. The extreme distribution with $a(n) = 34.5 + 4.2n$ and $b(n) = 9 + 3n$, where n is the number of players, is a good approximation for the server packet size. Exploiting this information, the cumulative distribution function (CDF) of the payload is shown in Figure 9, the top part showing a player having 5, 10, or 15 neighbors and the bottom part showing a single player. The client packets have an extremely narrow distribution with the mean size of 40 bytes [20]. Thus, an extreme distribution, with $a = 41$ and $b = 6$, is a good choice for the client packet size, leading to a packet size that is reasonable to carry the gaming functionality. For measurement purposes, we randomly assigned degrees to players between 1 and 4. Considering the attributes of modern workstations and the Internet, the chosen degree values are practical and reasonable.

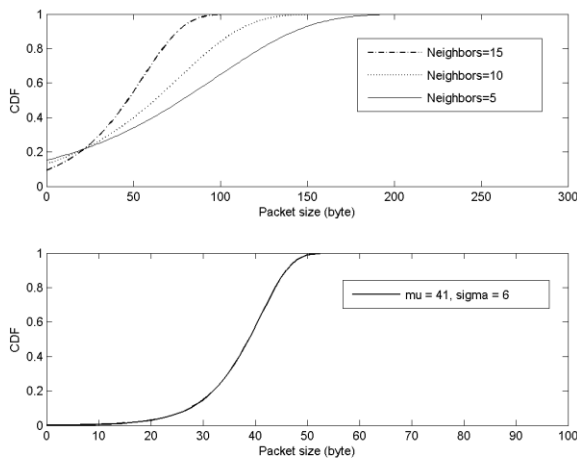


Figure 9. The cumulative distribution function (CDF) of the packet size (a) A Server for varying number of players (b) Client packet size

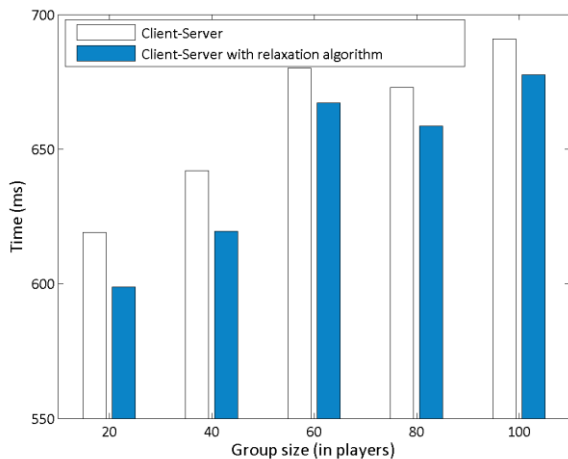


Figure 10. The drop-off of the maximum latecny due to the relaxation algorithm in client-server model

Usually, the size of an AoI is below 100 players (approximately 30 – 50). For example, World of Warcraft

supports Raid-groups² of up to 40 people, divided into 8 groups of 5 players. In this simulation, the players were placed randomly and the server was approximated to be in the middle of the players. For different group sizes (20 – 100), we measured the maximum latency of the client-server architecture. For the same settings, the proposed algorithm was applied to check its performance. The drop-off of maximum latency has been observed as can be seen in Figure 10. Thus, it is clear that the maximum latency through the server can be dropped and the players are likely to have a higher quality of gaming experience.

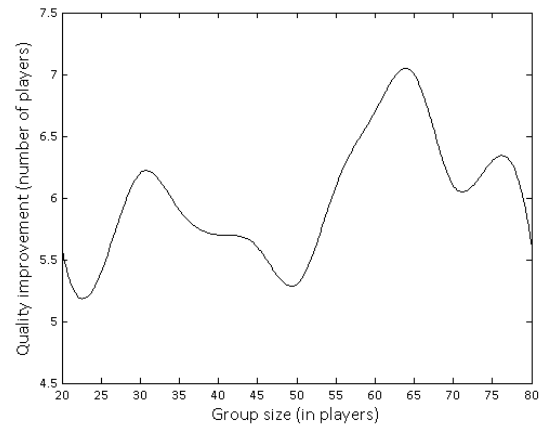


Figure 11. Improvement in maximum latencies for various group sizes

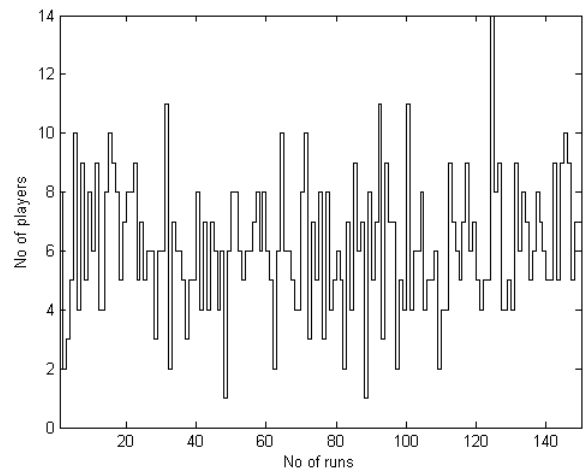


Figure 12. Quality improvement while consideing symmetric relationships

In addition, we also checked how many new players can now satisfy timing constraint for various group sizes after the use of the proposed concept. As the performance and improvement is tightly coupled with players' distribution across the Internet, the real picture was hard to determine. But the improvement is apparent according to Figure 11. For different group sizes, the quality can be improved for some players within the given degree limit and importance level (in our case, players had degrees between 1– 4).

² <http://www.wowwiki.com/Raid>

Finally, we also tested the role of the symmetric relationship on gaming performance. For a group of 20 players, we checked how many players' gaming quality is improved in terms of latency. Such a scenario for 150 runs is shown in Figure 12. The average improvement was 6.13 users for a group size of 20. For various group sizes, we also examined the overall quality in terms of the number of improved users as shown in Figure 13. For various group sizes, we counted the number of players where the quality was improved. The number increases with in an increment of group size as there are more options (i.e. more alternatives) to improve gaming experience. But it is a complex function of players' degree, inter-player relationship, physical locations, and game logic. By simulation, it is revealed that the consideration of temporal inter-player relationships while forwarding game states can improve gaming experience.

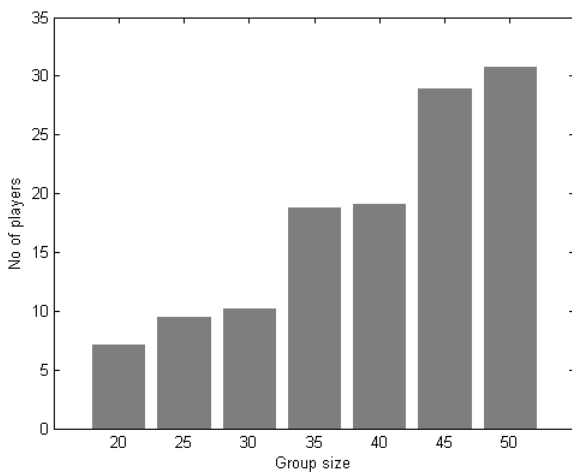


Figure 13. Quality improvement of various group sizes while considering symmetric relationships

VIII. CONCLUSION AND FUTURE DIRECTIONS

In this article, we have presented an approach for client-server architecture to satisfy the strict time-constraint of the target application through message redirection using participants' unexploited resources. In addition, we have proposed a quality control scheme for online games. Our assumption states that the closeness of interaction between two players is inversely proportional to their virtual distance. Based on this principle and time-constraint of the target application, a quality precedence matrix is formed; and if followed this can improve players' gaming experience. In adverse circumstances, the outlined procedure lets players share game states directly. The ultimate goal is to reduce latency and improve gaming experience of the players. The benefits are three-fold: (1) the time-constraint can be satisfied for more players (2) the load on the server is likely to drop or it can make room for more players with the same resources, (3) the higher importance can be given to the closer players for better gaming experience.

We also extended the game state sharing mechanism considering importance of players' transient interaction within

an AoI, and players' virtual and physical positions. Due to several variables such as players' movement, unstable and unpredictable interaction relationships, we cannot make an exclusive rule set for message exchange among players as a message exchange plan currently running has little value after a while. Considering these facts, we devise a message exchange plan that works alone in a client machine with the local information available. We have measured the performance, and the result shows clear benefits for games.

REFERENCES

- [1] S. Ratti, B. Hariri, and S. Shirmohammadi, "A Survey of First-Person Shooter Gaming Traffic on the Internet," *IEEE Internet Computing*, vol. 14, no. 5, pp. 60-69, 2010.
- [2] A. El-Sayed, V. Roca, and L. Mathy, "A survey of proposals for an alternative group communication service," *IEEE Network Magazine special Issue on Multicasting: An Enabling Technology*, vol. 17, no. 1, pp. 47-54, 2003.
- [3] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A Survey of Application-Layer Multicast Protocols," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 58-74, 2007.
- [4] D. T. Ahmed, S. Shirmohammadi, and I. Kazem, "Zone based messaging in collaborative virtual environments," in *IEEE International Workshop on Haptic Audio Visual Environments and their Applications (HAVE)*, 2006, pp. 165-170.
- [5] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation*. New York: SIGGRAPH Series. Addison-Wesley and ACM Press, 1999.
- [6] J. Smed, T. Kaukoranta, and H. Hakonen, "Aspects of networking in multiplayer computer games," in *International Conference on Application and Development of Computer Games in the 21st Century*, 2001, pp. 74-81.
- [7] M. Claypool and K. Claypool, "Latency and player actions in online games," *Entertainment networking SPECIAL ISSUE: Entertainment networking*, pp. 40 - 45, 2006.
- [8] J. L. Miller and J. Crowcroft, "The Near-Term Feasibility of P2P MMOGs," in *Proc. of International Workshop on Network and Systems Support for Games (NetGames)*, 2010.
- [9] G. Schiele et al., "Requirements of Peer-to-Peer-based Massively Multiplayer Online Gaming," in *Proceedings of the Seventh IEEE international Symposium on Cluster Computing and the Grid*, 2007, pp. 773-782.
- [10] J. M. Pullen, "Reliable multicast network transport for distributed virtual simulation," in *International Workshop on Distributed Interactive Simulation and Real-Time Applications*, 1999, p. 59.
- [11] S. Shirmohammadi and N. D. Georganas, "An end-to-end communication architecture for collaborative virtual environments," *Computer Networks*, vol. 35, no. 2-3, pp. 351 - 367, 2001.
- [12] IEEE standard for distributed interactive simulation - application protocols, 1998.
- [13] T. Fritsch, H. Ritter, and J. Schiller, "The effect of latency and network limitations on MMORPGs: a field study of everquest2," in *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, 2005, pp. 1-9.
- [14] M. Dick, O. Wellnitz, and L. Wolf, "Analysis of factors

- affecting players' performance and perception in multiplayer games," in *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, 2005, pp. 1-7.
- [15] M. Assiotis and V. Tzanov, "A distributed architecture for MMORPG," in *ACM SIGCOMM workshop on Network and system support for games (NetGames)*, 2006, p. 4.
- [16] T. Hampel, T. Bopp, and R. Hinn, "A peer-to-peer architecture for massive multiplayer online games," in *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, 2006, p. 48.
- [17] D. T. Ahmed and S. Shirmohammadi, "A Quality Control Algorithm Based on Virtual Distance in Games," in *International Conference on Embedded and Multimedia Computing (EMC)*, 2010.
- [18] S. Kaune et al., "Modelling the Internet Delay Space Based on Geographical Locations," in *Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2009, pp. 301-310.
- [19] C. Chambers, W.-c. Feng, S. Sahu, and D. Saha, "Measurement-based characterization of a collection of online games," in *ACM SIGCOMM conference on Internet Measurement*, 2005, p. 1.
- [20] W.-c. Feng, F. Chang, W.-c. Feng, and J. Walpole, "A traffic characterization of popular on-line games," *IEEE/ACM Transaction on Networking*, vol. 13, no. 3, pp. 488 - 500, 2005.
- [21] E. Manton, "Online Gaming: Where the Lost Boys Are," In-Stat Research Report IN0401178IA, 2004.