CONSISTENCY IN MULTIPLAYER ONLINE GAME IN CONTINUOUS DOMAIN

HPAI Pathirana[1] and RMCAB Rathnayaka
University of Vocational Technology
[1]pathirana@univotec.ac.lk

Abstract– This paper describes the ways to maintain the consistency in Multiplayer Online Games (MPOGs) in continuous domain. The involvement of computer networking, computer graphics, multimedia, artificial intelligence makes the MPOG environment more complex. As a result, consistency, responsiveness, scalability, fairness and avoiding cheating are examples for available issues in MPOG environment. The consistency is well discussed in the discrete domain, however the attention on consistency of MPOGs in continuous domain is poor and data centric approaches are not adequate for MPOGs due to the game state changes with passing time. Further, the consistency and responsiveness are two important considerations in the MPOG environment, however both cannot be achieved at once. As a result, the different architectures are available to implement game environment, and those are supported by latency handling techniques; time delay, local lag, time wrap, progressive slow down, dead reckoning, gossiping, PREMUB for example. The examples are discussed under each architecture for a better understanding of criticising the architecture. This paper mainly delivers the importance of consistency in MPOG, however it sacrifices the responsiveness in MPOG, as a result both aspects must be considered equally important for Quality of Experience (QoE) of the player.

Keywords- consistency, responsiveness, MPOG, latency compensation, QoE

I. INTRODUCTION

Multi-Player Online Games (MPOGs) are funded more and more by the entertainment industry, and those are getting popular increasingly. The one major reason would be the huge number of players in the internet for entertainment purpose, so it is easy to find a player at any time. The board games, card games, turn based games, word games, role playing games, Role Playing Game (RPG), Real Time Strategy (RTS) games, sports games and First Person Shoot (FPS) games are example for MPOGs, however it is not possible to treat them in same manner due to the different consistency and responsiveness requirements.

Turn based board games, card games are examples for games in discrete domain, since state updates happen as discrete event based on the user initiated operation; no relation of the passing of time. Further, there is no time constrain to present a state of game in discrete domain. These games barely satisfy the requirement of MPOG by maintaining interactive game online, further the inherent causality preservation of games in discrete domain is enough for maintaining consistency in such games. This paper does not focus on such game environments.

In general, MPOG is a type of distributed real time application. RPG, RTS, FPS and sports game are good examples to understand the nature of MPOGs. The document type for interactive MPOG is dynamic, because game state changes for two reasons; relation of the player initiated operation, and the passing of time. The research in consistency maintenance for replicated dynamic document is relatively under explored with reference to the replicated static document.

There are big collection of Massively Multiplayer Online Games (MMPOG) as per the MMPOG Dictionaries online (MMOG Dictionary, 2017; MMORPG, 2017) and those convince the present demand for playing MPOG. For example, the site shows 3686804 members, 974 online users and 952 games on 16th October, 2017 (MMORPG, 2017). MPOG is an opportunity for someone to play game at any time with someone else online. The great level of MPOG collection provides choice for the player.

The MPOG has two platform in general; physical platform and logical platform, whereas the existing physical platform must be supported by the logical platform as the goal of best performance (Hsu, Ling, Li, & Kuo, 2003). In other words, a player represents the physical platform, and a virtual site represents the logical platform. The logical state/virtual site is mainly for consistency maintenance.

The consistency is about the same game state in all instances of the players, whereas responsiveness is concerned on delay for an update to register throughout the network (Gao, Jin, Shen, & Babar, 2017). A shooting dead man is an example for consistency problem; player had shot a man to dead that is responsive enough to reflect locally to determine dead man, however remote player use that dead man to shoot continuously till update is reflected. Highly interactive game environment requires higher responsiveness. Both are important to consider for MPOG environment, however both are are contradictory goals oftenly. As a result, one gets priority over the other depending on the nature of specific MPOG environment. Thresholds for the responsiveness are FPS – 100ms, RPG – 500ms, RTS – 1000ms for example (Gao et al., 2017), more importantly it concludes FPS focus more on consistency over RPG, RTS.

The latency is an inherent major problem to maintain consistency for MPOGs, since multiple players are connected over heterogeneous network; the Internet. Latency is further crucial, as responsive threshold for the MPOG should be within Round Trip Time (RTT) of two game instances (Li, Li, & Lau, 2004). For example, the Final Fantasy XI has discussed in the same paper as one of the popular MPOG in 2004 for example, there is almost one second of delay for having position updates from the other players. Moreover, the restrictions had introduced to reduce those delays to maintain consistency; attack only on objects of enemy, controlled moving speed for objects, but those limits on expected features of the game.

Moreover, the jitter is equally important issue for MPOGs. It happens due to the latency variation. The one common approach on treating jitter in MPOGs is based on maintain threshold for the jitter; farcing to leave the game avoiding counter effects. However it influences on the game fairness and the player QoE negatively (Gao, Shen, & Babar, 2016), and Predictive Modelling of User Behaviour (PREMUB) has been introduced for seamless transmission between player and its intelligent agent as alternative approach. It allows maintaining consistency against poor response.

The rest of this paper further discussed above matters in MPOG environment as follows. The consistency requirements for MPOG environment are disused in detail at next; section 2. Then, the latency handling techniques are discussed as solutions for inconsistencies in two aspects; time based solution, predication based solution in section 3. In section 4, available different architectures are discussed with respect to the consistency concerns with practical example. Finally the paper is concluded with important findings and future directions.

II. CONSISTENCY IN CONTINUOUS DOMAIN

The strictly consistent is about identical replicas at all the time; it is an ideal case. Due to the inherent communication latency among collaborating sites, the strict consistency can never be achieved. There is delay always for propagation from originated game instance to other game instances (Gao et al., 2016; Li et al., 2004; Zhang, Kemme, & Denault, 2008; Zhou, Cai, Lee, & Turner, 2004). Moreover, the human user is able to tolerate a temporary divergence among replicas; as a result the applications do not expect to maintain strict consistency.

For MPOG, due to the dynamic nature of the document, it crucially important to maintain consistency at greatest level; not like discrete domain which focus on correct sequence of independent operation at each site. The consistency of application in continuous domain is focused on execution of operation at correct point in time essentially in addition to the execution in correct sequence as in discrete domain (Mauve, Vogel, Hilt, & Effelsberg, 2004). As a result, discrete domain algorithms to maintain consistency are not sufficient enough for continuous domain to maintain consistency; however those algorithms in discrete domain to maintain consistency are the foundation into continuous domain for introducing relevant algorithms for continuous domain. Further, the continuous domain consistency criteria can thus be regarded as a specialization of the use of the discrete domain consistency criteria.

More interestedly, a single operation is adequately enough for introducing inconsistency for highly interactive MPOG, because there is communication latency in between sites for propagation (Shen & Zhou, 2013). Latency is main hindrance for consistency and responsiveness, and it introduces when the responsiveness threshold of a game cannot be fulfilled within the RTT.

If all the players have received all the operations at any given time, it is ensured consistency with identical state in all the sites at that time (Mauve et al., 2004). However if all the sites has not received all required information at any given time, it is not possible to conclude the state of consistency at given time. Nevertheless, The synchronization of the distinct physical clocks are not able to assist on consistency in continuous domain (Mauve et al., 2004). The same readings of the physical clocks of the all players are important for consistent criteria, if all previous operations have already been received. The same reading of a common wall clock is irrelevant, even though consistency is achieved or not.

The correctness, short-term inconsistency and responsiveness are terms attached with consistency, and those are discussed further in this section at next.

A. Correctness

Beside consistency, the correctness of all the game states is important among all the states of the sites of application in continuous domain (Mauve et al., 2004). The correctness is completely independent of the synchronization of physical clock at distinct sites in theory, however there is no practical issue due to the limitation of human resolution perception as long as the calculation of state is fast sufficiently.

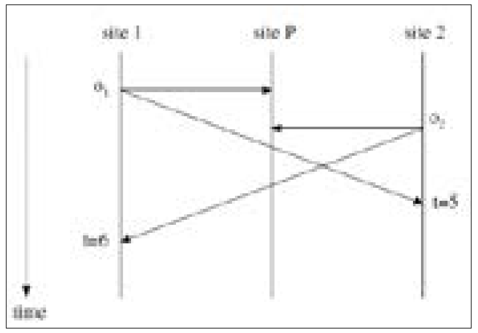

Figure 1. Consistency and Correctness (Mauve et al., 2004)

The consistency and correctness have been described for example as in Figure 1 (Mauve et al., 2004). Site 1 and Site 2 are real to represent a separate instance of continuous distributed application; game instance of player. Site P is virtual site which receives all the user initiated operations; vertical arrow from real site to virtual site describes no delay for updating virtual site. The idea behind the diagram is that the correctness is checked always with the virtual sites which execute operation in order; the correctness has been achieved for identical states in both site instance state and virtual site state. In fact, the consistency is guaranteed when actual sites are identical. As a result, there are situation where consistency is assured, however correctness is not assured. It is important to remember, the correctness is vitally important for MPOG environment on top of the consistency.

B. Short-Term Inconsistency

The short-term inconsistency occurs in a situation where minimum of two sites have different states. It is very common concern, since there is propagation delay from the originated site of the operation to the one other site to appear the operation, so it does not refer the violation of either constancy or correctness under the assumption of primary requirement of application to handle such situation (Mauve et al., 2004). The transmission delay, offset between the physical clock, time gap between operations issued and operation executed are the factors to decide the short-term inconsistency. For example, in Figure 1 the short-term inconsistency appears in site 1; from the time O2 issued to t=6, and site 2; from the time O1 issued to t=5. In the MPOG, this situation is handled by the game software up to some extent, even though it should not be existed in ideal case.

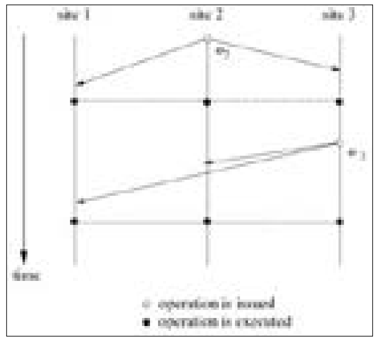

Figure 2. Minimizing Short Term Inconsistencies (Mauve et al., 2004)

The short-term inconsistencies have been minimized in Figure 2 by executing an operation after some time, response time, at the same scheduled time instead of executing at the time of operation issued at the originated site. The response time is estimated assuming the propagation delay of the operation on the other sites, as result it is guaranteed all the participating sites have been received the operation to execute. Further, it is desirable to reduce short-term inconsistency on the cost of responsiveness, while it is not guaranteed on prevention of short-term inconsistency.

i. Game Responsive Threshold

In MPOG, short-term inconsistency can be handled assuring no influence on experience of the player. Due to the different nature of the games, there is specific threshold value to maintain for response time which is game responsive threshold, even though the higher response time can be introduced to have greater emphasis on consistency. There are two groups of MPOGs under this criterion: game either with higher responsive threshold or low responsive threshold.

The higher responsive threshold is applicable for RPG and RTS inclines to have emphasis on consistency, and the lower responsive threshold is applicable FPS and sports game needs greater weight on responsiveness. As a result, short-term inconsistency is not acceptable for games with lower responsive threshold or lower game responsive threshold and there should be other techniques to maintain consistency.

C. Responsiveness

The response time of continues replicated application should be within certain threshold, otherwise local user experiences an unnatural delay in their local site (Mauve et al., 2004). Figure 3 shows site 2 and site 3 issue O2, O3 and those are executed with no response time to assure experience of local user; ideal case. The responsiveness has been optimized in that case at local site, however short-term inconsistency is appeared.

More importantly, the both fidelity criteria; short-term inconsistency and responsiveness, are conflicting goals, because more attention on one criteria influences on the other criteria. In MPOGs, responsiveness is achieved in acceptable level.

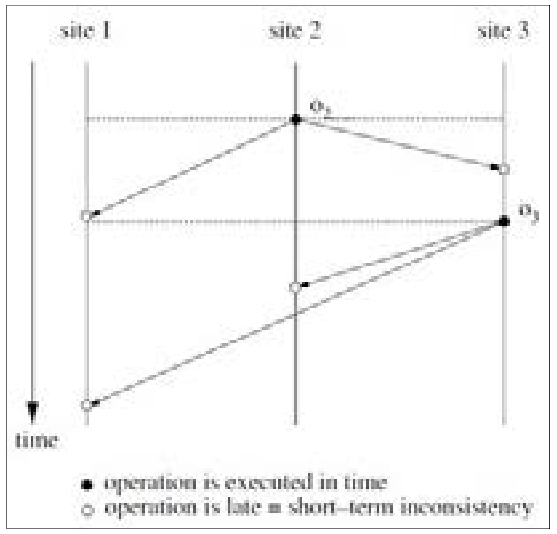
site 1
site 2
site 3
o2
o3
time
● operation is executed in time
○ operation is late = short–term inconsistency

Figure 3. Optimizing Responsiveness (Mauve et al., 2004)

i. Replication

Although the operation is executed without waiting, it is not guaranteed on responsiveness due the inherent communication latency (Mauve, 2000; Mauve et al., 2004). It influences on appearing effect of an operation in one site to appear among other multiple sites for highly interactive MPOG. Nevertheless, there is issue on responsiveness at the local site also with client-server, cloud implementation. The issue at the local site is addressed by maintaining full or partial replica of game state locally, and it is default approach of peer-to-peer approach. It addresses the responsive problem, however it introduces the consistency problem.

D. Causality Preservation

In the continuous domain of MPOG, this introduces interesting problems; due to the short-term inconsistency; if there is no short-term inconsistency, causality is perfectly preserved, since operation executes on all the sites at correct point in time (Mauve, 2000; Mauve et al., 2004). The short delay allows one operation to lead an old concurrent operation. It is essential to delay the new operation until the old concurrent operation to appear and execute. As a result, that new operation introduces short-term inconsistency on future operations in undesirable manner in interactive continuous domain (Mauve et al., 2004). Due to that, the consistency and correctness criterion are not considered on causality reservation in continuous domain. In the MPOG environment, the causality preservation is not be able to assure due to the above endless delays to assure consistency, as a result short-term inconsistency is not addressed through causality preservation techniques.

III. LATENCY HANDLING TECHNIQUES

Latency is an inherent problem and major hinderance for MPOG environment, because player feel unnatural behavior due to latency (Mauve et al., 2004). The many solutions have been introduced based on assumption on acceptable threshold. Either consistency or responsiveness is focused by the existing solutions. The solutions are based on two aspects; time based approach and predication based approach. Moreover, combination of those techniques is also available. Those multiple solutions are discussed under each category in this section at next.

A. Time Based Solutions

The time based solutions are focused on maintaining consistency, as a result responsiveness impacts negatively.

i. Time Delay / Lockstep

The user commands are not executed immediately, instead those commands are hold by the server for a short period of time, and all the commands are executed in order at the end (Shen, 2017). As shown in Figure 4, client 1 and client 2 send commands to server at different point in time in the timeline, however server holds with the operations of both client to process and execute without introducing inconsistencies and fairness of the game is improved. Unfortunately it introduces matters on responsiveness.

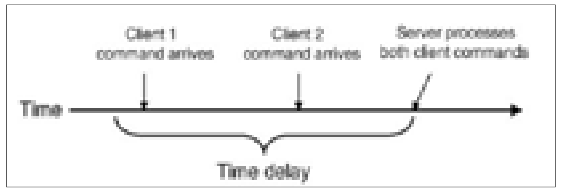

Figure 4. Time Delay (Shen, 2017)

The lockstep is stop-and-wait type approach, which helps on maintaining consistency on the cost of responsiveness(Baughman & Levine, 2001). It is extension for time delay approach with cryptographic approach for avoiding cheating. Further, this approach is not applicable for highly interactive MPOG environment due to the poor responsiveness, even though consistency is maintained.

ii. Local Lag

The local lag has been proposed to address the tradeoff relationship between response time and short-term inconsistencies. The operation is not executed immediately after it is issued by the local user (Mauve, 2000; Mauve et al., 2004), however this concept focus on reducing the responsiveness of the medium. The value for the delay for introduce local-lag must be chosen in the way of neither noticeable nor distraction. Moreover, this approach is not just delaying the operation to execute, but it is all about tradeoff among short-term inconsistency and responsiveness. Finally, it is essential to understand, the local lag sacrifices the responsiveness for better consistency. For example, in Figure 2, the O2 in site 2, and O3 in site 3 have been delayed after they are issued for execution; local lag, to assure consistency.

A minimum and maximum values for the local lag, as per the application tolerance, are determined to finalized the fair value for the local lag for an application based on network delays, jitter, probabilities etc. literature suggest to use 80-100 ms as local-lag with no influence on the user considering even psychological aspects. Finally, the consistency is preserved with fair delay to execute at local site without influencing on responsiveness severely. This approach is not recommended for highly interactive MPOGs due to the poor responsiveness even though consistency is preserved.

iii. Time Wrap

A sufficiently large local lag is able to eliminate short-term inconsistency significantly, but it is not ultimate solution for that due to the jitter, losses over the network (Mauve et al., 2004). As a complement local lag, there should be mechanism to assure consistency and correctness, and time wrap is addressed that problem, however it influence on responsiveness.

Time wrap algorithm waits for predefined period of time to collect local operations and remote operations to store during that short period (Mauve et al., 2004). That short period is based on the requirement of the application based on frequency

to calculate new state. In MPOGs, server keeps some snapshots of every recent game states of players with the timestamp (Mauve et al., 2004). If it is necessary to have 25 updates with in a second, the short period of time is calculated as 40 ms. The reconstruction of any inconsistency is conducted with the help of stored interrelated details until state reaches to the current time. The correctness is also assured through complex set of steps with the help of virtual site.

In MPOG, if one player's command involves another player server calculates both player's latency, roll back game state to the previous state of the act moment player sent the command (Mauve et al., 2004). The downside in this approach is the complexity for the computation. The assumption on providing correctness in the time wrap approach reduce the complexity, however using time wrap is a challenge for real time application like highly interactive MPOGs. As a result the complexity of time wrap approach takes into account for giving a solution.

iv. Progressive Slowdown Latency Compensation

This approach addresses both consistency and responsive matters, and this solution only for highly interactive online ball game with two-player (Shen & Zhou, 2013). It includes critical consistency model which expect to have the consistent view at critical state of the game, and it uses this approach for achieving critical consistency. The consistency view is expected have at the critical region as shown in Figure 5, and it is allowed to have inconsistent view at non-critical region. This approach is specific for two player ball games, and each player must have a turn to hit the ball.

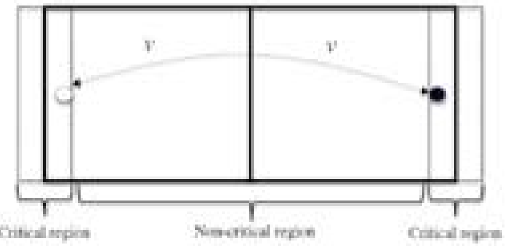

Figure 5. A highly Interactive Two Player Ball Game Court (Shen & Zhou, 2013)

As shown in Figure 6, there are different speeds for the ball as at different location. At the time, ball is close to the player, the speed is maximum for the smoothness and naturalness of the game. The speed is progressively reduced for the inherent latency baring due to the communication for QoE of player, when ball goes away further from the player. The online table tennis game has been use to demonstrate this scenario (Shen & Zhou, 2013). The outcome is satisfied and they are focusing on enhance this for multiple players.

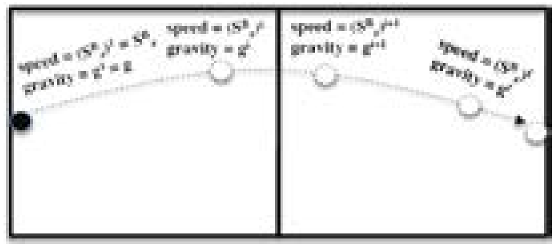

Figure 6. The Progressive Slowdown method for Latency Comparison (Shen & Zhou, 2013)

This method is applicable for peer-to-peer two player online ball games, but not more than two player.

B. Prediction Based Solutions

The state is able to predict, even though there is loose of data. The aim of prediction based approach to reduce responsiveness not giving priority for consistency.

i. Dead Reckoning

Dead reckoning uses to assure, that the consistency criterion is satisfied in distributed virtual environment (DVE) (Mauve, 2000). Both state prediction and state transmission are related with dead reckoning. The behavior of the object in the DVE over the time must be known by the application for this solution. As example, a plane flies in one direction for constant speed or in a given line as calculated considering gravity (Mauve, 2000). The capability to forecast the behavior of an entity is called dead reckoning.

The application maintain single controlling instance at every time for the objects belongs for dead-reckoning (Mauve et al., 2004). The application instances of the pilot are an example from the plane scenario. The controlling application only has capability to change the object state, when the state of object moves over more than a given threshold value from the predicted state. Then control application sends the affected object related complete state to all other states.

The dead reckoning is able to facilitate consistency, however it is unable to guarantee correctness, because information of the events is not shared with other sites, instead state information is shared (Mauve et al., 2004). Moreover, it is not recommended to send information of the events due to the unreliable transmission. Further this may overload the network if all the sites are going to correct the states, since no reference to follow as correct version; all sites maintain relative state.

The responsiveness is also handled accordingly in dead reckoning. In the game environment, local state is updated with a fixed time delay (100 ms) for providing state update time to reach destination. The choice of time warp for maintaining consistency in the use of dead reckoning has not recommended due to some further implication; responsive might not be able to address.

For the MPOGs, correctness is utterly important to prevent cheating and for fairness. Therefore, it is necessary to address above maters to adopt dead reckoning for MPOGs. Moreover, time wrap would be much appropriate for heavily interactive MPOGs over the state sharing in dead reckoning.

ii. Speculation

This approach predicts future frames which can be possibly appear within RTT (Lee et al., 2015). The state prediction, state approximation, checkpoint and rollback for state, compression state or saving network bandwidth are related concept under this approach to achieve better responsiveness, however the consistency is not addressed.

The Outatime Architecture addresses the latency issue at the server end (Lee et al., 2015) based on speculation concept. Outatime facilitates to hide 120 ms of latency over Internet in the mobile cloud gaming, and it extracts speculative frames to represent potential outcome at next, and transmitting those frames to the other sites before one complete RTT of time (Lee et al., 2015). It helps on recovering wrong/missed speculations. Future state prediction, state approximation with image based rendering and event time shifting, fast state checkpoint and rollback, state compression for bandwidth saving are steps of Outatime. The demonstration had

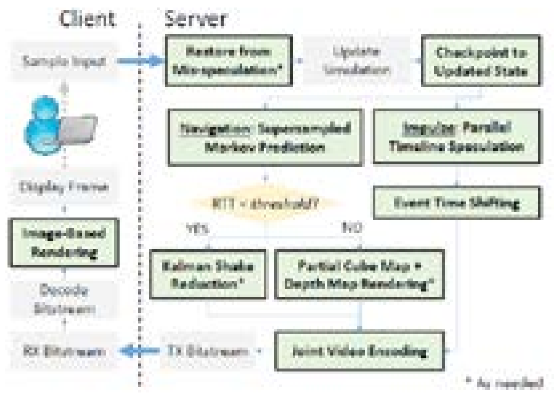

Figure 7. The Outatime Architecture (Lee et al., 2015)

conducted for Outatime on Doom3 FPS game, and it concludes that game providers can implement this for better user experiences.

C. Other Approaches

There are other approaches as combination of both time based and state prediction; gossiping (Lu, Parkin, & Morgan, 2006) and Obsolescence-based Optimistic Synchronization (Shen & Zhou, 2013). Nevertheless, a new approach has introduced with the use of an intelligent agent locally to represent remote user; humanoid bot.

i. Predictive Modelling of User Behaviour (PREMUB)

Intelligent agent is used to model the human behavior in this case, then the humanoid can be used to represent human behavior for any inconsistency (Gao et al., 2017). It improves QoE of the player. The challenge here is that humanoid bot is able to represent the player when there is no inconsistency to imitate the player, however humanoid bot is effective to use against latency hike, known as jitter, instead of leaving the player out.

There is concept to switch for humanoid bot for jitter (Gao et al., 2017). There is no issue when latency is below the threshold of a MPOG, and actual player continue on playing under that situation. There is a background process while continuing normal scenario, each player share its gameplay data with remote site to introduce humanoid bot for them at remote site. At some point in time, latency may exceed the threshold of the game, then play stops communicating with remote site, instead player communicates with humanoid bot of remote site which

is already established locally. It provides time for latency to become value lower than threshold of the game, then players synchronize the states to start playing game, once player starts the game, humanoid bots go into standby mode.

This is a very new approach to improve satisfaction of the player through seamless background operation. In the case study, the pong game has used as MPOG to evaluate this approach, and it is concluded, that there is some promising results on behavioral variables, process, modelling techniques, performance and outcome (Gao et al., 2017), however their verification process is based on historical gameplay data single test subject for short period of time.

IV. MPOG NETWORK ARCHITECTURES

The involvement of interactive computer interfaces, heterogeneous networking infrastructure, hardware support, artificial intelligence and number of players makes the MPOG environment complex. In the literature, the large scope of MPOG provides some guidance for available issues on addressing consistency and responsiveness requirements instead of providing exact solution. The ultimate goal of an architecture is to deliver specific MPOG environment that player is comfortable to use by maintaining consistency and responsiveness; QoE.

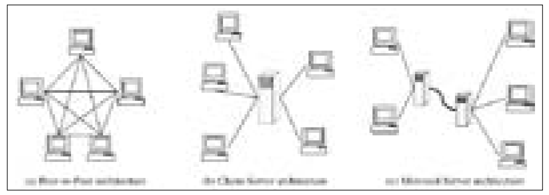

Figure 8. Topology for Multiplayer Online Game System (Hsu et al., 2003)

A. Client-Server Architecture

The Figure 8 – (b) illustrate the topology for this, and the server entity makes decisions. The client-server architecture is better approach for MPOG, since server assist on having consistency among all the playing instances (Hampel, Bopp, & Hinn, 2006). It is based on unicast TCP; a connection oriented lossless transmission is guaranteed. The server maintains the state of the game state applying transformation as required, and each client handles input, output and collision detection for transformation. It favors for consistency ahead of responsiveness; sacrifices the responsiveness to maintain consistency. However client-server communication introduces a latency influencing the user experience due to the poor performance of network communication. Nevertheless, the MPOG industry focus on client-server approach for commercial objectives to facilitate massive number of players at once (Lu et al., 2006).

In general, the client-server architecture is recommended for having data-centric consistency and appropriate for identified MPOG type; no state change against time, acceptable latency in client-server communication (Shen & Zhou, 2013). The client-server is not recommended for more interactive MPOG, because the game state is changed with related to the time progress. For example, P1, P2 are two players of pong game in Figure 9. The local response for P1 has been extended due to the involvement of the server, and the remote response has also been extended compared to peer-to-peer architecture scenario shown in Figure 10. It is very crucial to maintain responsive time within pong game responsive threshold to stay with the game. The server performance is also influence on responsiveness due to the massive number of players.

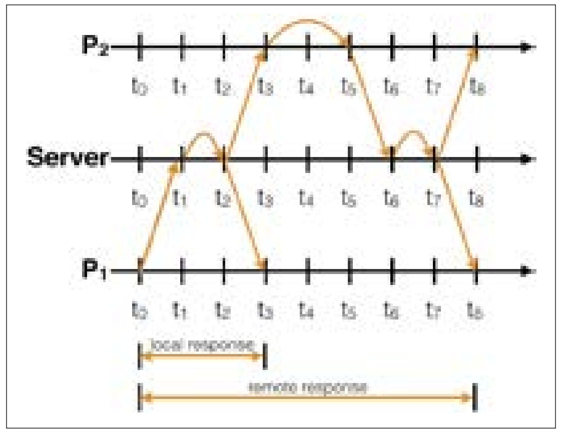

Figure 9. Impact of Latency on Responsiveness (Gao et al., 2017)

i. Distributed Server

The multiple servers are contributed for MPOG, because the single server representation is a bottleneck for MPOG environment (Lu et al., 2006), and the Figure 8 – (c) illustrates mirrored server architecture with additional server. The influence due to the physical distance of the player is also addressed in multi-server; reduced latency with closest server communication, and server must be connected reliably with a fast connection for better synchronization among them. The responsiveness is enhanced compared to single server environment, and consistency is depend on the consistent communication between servers mainly; if they far away still consistency is issue, even though it is better than having single server.

Further, massive number of players must be supported with cluster of servers utilizing cumulative resources for processing and maintaining (Lu et al., 2006). The clustered-server architecture assists on load balancing. If one server is down due to some reason, there are some other servers nearby to take over the responsivity seamlessly. Clustered server environment is far more supportive for maintaining consistency, whereas responsiveness may also improve depending on the distance between player and server closer the distance is much better.

B. Peer-to-Peer Architecture

The peer-to-peer architecture represents a communication between two players connected somehow. It maintains fully replicated game state, and there is direct communication to maintain consistency of replicas (Diot & Gautier, 1999; Ferretti, 2008). The Figure 8 – (a) illustrate the topology for this, and all the participating entities make decisions. It favors for responsiveness ahead of consistency. As a result, inconsistency appears, and repair inconsistency by using latency-hiding techniques discussed above. So, consistency, responsiveness, scalability, avoiding cheating and fairness are example for challenges in this architecture.

On the other hand, MPOGs in peer-to-peer use Real Time Protocol (RTP) over UDP/IP; data can be loose due to the connectionless unreliable transmission. No central server exists to maintain state of game, as a result each participating entity maintain own game state resulting the greater possibility of inconsistency. It is responsibility of each entity to contribute for synchronization between multiple game states with heterogeneous Internet delays.

The following figure discusses impact of latency using a pong game. There is no any influence on local response since event of P1 reflects locally with no delay, however remote response still has influence, though it is better than client-server architecture. It is crucial to maintain response time within pong game responsive threshold to stay with the game, however that risk is low compared with client-server environment. It further concludes, that it is necessary to maintain different threshold for different architecture in the cost of responsiveness.

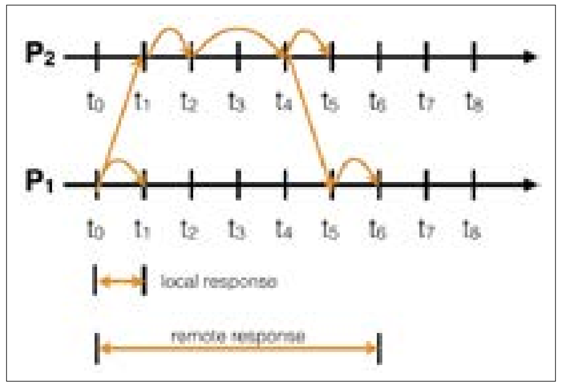

Figure 10. Impact of Latency on Responsiveness (Gao et al., 2017)

Considering the local response time and remote response time among client-server architecture and peer-to-peer architecture, the choice for highly interactive MPOG is peer-to-peer architecture in general, but the consistency can be easily maintain in client-server environment with massive number of player compared to peer-to-peer architecture.

i. Server Mediated Peer-to-Peer Architecture

The disadvantages of peer-to-peer architecture due to the unstructured and decentralized environment; no central coordination, are addressed in this approach (Kwok, Chan, & Cheung, 2005). The server mediated approach has introduced for addressing the consistency issue fair sacrificing on responsiveness. The all nodes in game environment are not players, some nodes represent the role of server as a facilitator/coordinator. In MPOG, even though the server facilities the game environment, that server does not contribute on decision making (Shen, 2017). The inter-trusted peers are grouped around a server for sharing networking and computing capabilities for better experience of the player.

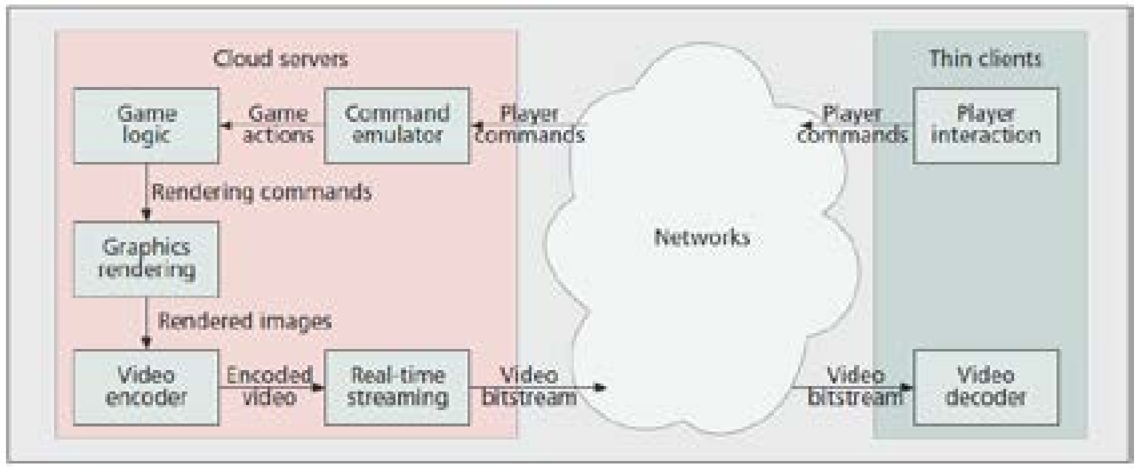

Figure 11: A Cloud Gaming Framework (Chuah et al., 2014)

MiMaze is an example for implementation of such architecture (Diot & Gautier, 1999). It does not maintain a central server to game state, however it uses sever for non-real-time task such as session management for new player. The distributed games must maintain consistency, so it is essential to implement synchronization mechanism at minimum. The acknowledgment based approaches introduces additional communication to the busy Internet (Diot & Gautier, 1999), however dead reckoning algorithm to recover losses and latency of packets has been considered.

C. Cloud Games Architecture

The cloud infrastructure heavily supports for MPOGs on portable devices; smart phones, laptop, tablet (Ferretti & D'Angelo, 2012; Lee et al., 2015). The remote servers is responsible to perform execution of game and rendering as per the input from the client, and thin client displays output frame. Further, it facilitates to play any game at any time ignoring the performance of the thin clients by delegating complex computation into cloud end (Chuah, Yuen, & Cheung, 2014). Further, it helps to save the battery of mobile devices for better user experiences.

Cloud game provides less maintenance, no upfront cost, better scaling (Chuah et al., 2014). Game developers are also further benefited due to the pay as you go concept, since they no need to pay upfront cost to introduce game, however the growing number of players needs more resources which can be allocated instantly. These motivates players to play MPOG and developers to develop games. The MPOG is going to be very big industry.

In Figure 11, a MPOG framework has introduced to illustrate video game environment over the cloud (Chuah et al., 2014), it shows clearly the contribution of cloud server for thin client; all the processing and computations have been delegated into the cloud server.

The massive number of players involvement on MPOG is complex to predict resource requirement to accommodate all the users with better gaming experience, and resource constrains lead for inconsistency, as a solution cloud facilitates scalable resource with no interruption (Chuah et al., 2014). The cloud is very useful to implement the game environment by introducing required physical infrastructure for any architecture delegating complex computation, resource requirement for spike of players, maintaining consistency. Cloud game architecture maintain consistency much better way, and responsiveness depends on the network delay in the Internet access.

V. CONCLUSIONS

The existing literature is reviewed for evaluating consistency of MPOG in continuous domain. . The involvement of computer networking, computer graphics, multimedia, artificial intelligence makes the MPOG environment more complex. But MPOGs are increasingly popular with the interest of the players and the affordable infrastructures requirement; mobile devices with better performance, countless cloud services, wideband of the Internet facility. However, highly interactive MPOGs have challenges due to the network latency, and it impacts on design and deployment of highly interactive MPOG. The latency handling techniques are used to address issues as shown in Table 1. The requirement of MPOG decides the most relevant approach. Finally latency must satisfy the game threshold to assure QoE of player. Further, different MPOG network architectures are discussed for understanding the support to maintain consistency requirement in MPOG environment.

Table 1. Summary of Latency Handling Techniques

Techniques	Time Manipulation	Prediction Based	Responsiveness	Consistency
Time Delay	x			x
Local Lag	x			x
Time Warp	x			x
Progressive Slowdownx		x	x	
Dead Reckoning	x	x	x	
Speculation	x	x		
PREMUB		x	x	

In summary, the available latency handling techniques and network architecture address consistency requirement of highly interactive MPOGs, however certain solutions only work well with certain architecture as shown in Table 2. The table has been introduced based on present personal understanding on discussed different architectures and latency handling techniques in this paper, however specific considerations should be based on actual MPOG environment requirement instead of architecture to adopt best latency handling technique. Finally, interactive MPOG still suffer due to the consistency issue. It is necessary to conduct comprehensive research focusing highly interactive MPOG approaches.

Table 2. Summary of use of Latency Handling Techniques

MPOG Network Architecture	Time Delay/ Lockstep	Local Lag	Time Warp	Progressive Slowdown	Dead Reckoning	Speculation	REMUB
Client-Server			x		x		
Distributed Server			x		x		
Clustered Server			x		x		
Peer-to-Peer			x	x	x	x	x
Server Mediated P2P			x		x	x	x
Cloud Games					x		

REFERENCES

Baughman, N. E., & Levine, B. N. (2001). *Cheat-proof playout for centralized and distributed online games.* Paper presented at the INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE.

Chuah, S.-P., Yuen, C., & Cheung, N.-M. (2014). Cloud gaming: a green solution to massive multiplayer online games. *IEEE Wireless Communications*, 21(4), 78-87.

Diot, C., & Gautier, L. (1999). A distributed architecture for multiplayer interactive applications on the Internet. *IEEE network, 13(4)*, 6-15.

Ferretti, S. (2008). A synchronization protocol for supporting peer-to-peer multiplayer online games in overlay networks. Paper presented at the Proceedings of the second international conference on Distributed event-based systems, Rome, Italy.

Ferretti, S., & D'Angelo, G. (2012). *Mobile online gaming via resource sharing.* Paper presented at the Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques, Desenzano del Garda, Italy.

Gao, C., Jin, K., Shen, H., & Babar, M. A. (2017). Are you a human or a humanoid: Predictive user modelling through behavioural analysis of online gameplay data. *Advanced Engineering Informatics*.

Gao, C., Shen, H., & Babar, M. A. (2016). *Concealing jitter in Multi-Player Online Games through predictive behaviour modeling.* Paper presented at the Computer Supported Cooperative Work in Design (CSCWD), 2016 IEEE 20th International Conference on.

Hampel, T., Bopp, T., & Hinn, R. (2006). *A peer-to-peer architecture for massive multiplayer online games.* Paper presented at the Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games, Singapore.

Hsu, C.-C. A., Ling, J., Li, Q., & Kuo, C. (2003). *The design of multiplayer online video game systems.* Paper presented at the Proceedings of SPIE.

Kwok, S. H., Chan, K. Y., & Cheung, Y. M. (2005). A server-mediated peer-to-peer system. *SIGecom Exch.*, 5(3), 38-47. doi:10.1145/1120680.1120686

Lee, K., Chu, D., Cuervo, E., Kopf, J., Degtyarev, Y., Grizan, S., . . . Flinn, J. (2015). *Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming.* Paper presented at the Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services.

Li, F. W., Li, L. W., & Lau, R. W. (2004). *Supporting continuous consistency in multiplayer online games.* Paper presented at the Proceedings of the 12th annual ACM international conference on Multimedia.

Lu, F., Parkin, S., & Morgan, G. (2006). *Load balancing for massively multiplayer online games.* Paper presented at the Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games.

Mauve, M. (2000). *Consistency in replicated continuous interactive media.* Paper presented at the Proceedings of the 2000 ACM conference on Computer supported cooperative work.

Mauve, M., Vogel, J., Hilt, V., & Effelsberg, W. (2004). Local-lag and timewarp: providing consistency for replicated continuous applications. *IEEE transactions on Multimedia*, 6(1), 47-57.

MMOG Dictionary. (2017). MMOG Collection. Retrieved from http://mmogdirectory.com

MMORPG. (2017). MMORPG Collection. Retrieved from http://www.mmorpg.com/

Shen, H. (2017). COMP 7722 - *Computer Supported Cooperative Work and Groupware.* Retrieved from Flinders University - Adelate - Asustralia

Shen, H., & Zhou, S. (2013). *Achieving critical consistency through progressive slowdown in highly interactive multi-player online games.* Paper presented at the Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on.

Zhang, K., Kemme, B., & Denault, A. (2008). *Persistence in massively multiplayer online games.* Paper presented at the Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games, Worcester, Massachusetts.

Zhou, S., Cai, W., Lee, B.-S., & Turner, S. J. (2004). Time-space consistency in large-scale distributed virtual environments. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 14(1), 31-47.