VFC-RTS: Vector-Field Consistency para Real-Time-Strategy Multiplayer Games

Manuel Eduardo Costa Cajada

INESC-ID/Instituto Superior Técnico
Distributed Systems Group
cajadas@gmail.com

Adviser: Professor Luís Manuel Antunes Veiga, (DEI/IST)
Co-adviser: Professor Paulo Jorge Pires Ferreira, (DEI/IST)

Abstract

Although massive multiplayer online games have been gaining most popularity over the years, real-time strategy (RTS) has not been considerate a strong candidate for using this model because of the limited number of players supported, large number of game entities and strong consistency requirements.

To deal with this situation, concepts such as continuous consistency and location-awareness have proven to be extremely useful in order to confine areas with consistency requirements. The combination between these two concepts results on a powerful technique in which the player's location and divergence boundaries are directly linked, providing the player the most accurate information about objects inside his area-of-interest.

The VFC model achieves a balance between the notions of continuous consistency and location-awareness by defining multiple zones of consistency around the player's location (pivot) with different divergence boundaries.

In this work we propose VFC-RTS, an adaptation of the VFC model, characterized for establishing consistency degrees, to the RTS scenario. We describe how the concepts of the original VFC model were adapt to the RTS paradigm and propose an architecture for a generic middleware. Later, we apply our solution to an open source, multi-platform RTS game with full consistency requirements and evaluate the results to define the success of this work.

1 Introduction

With the increase of the internet services capabilities over the years there has also been a growth in the new types of real-time applications. One that has been gaining most popularity are massive multiplayer online games (MMOG).

The category most played and with the greatest popularity are the role-playing games (MMORPG), however, there are other real-time network game categories which have been less discussed for using the MMOGs model, among them the real-time strategy games (RTS)[7].

Multiplayer solutions for RTS games typically allow a limited number of participants to be playing simultaneously[5] due to the lack of scalability of the traditionally used network architecture, the Client-Server (CS) model. This architecture is characterized by the use of a centralized server that is responsible for maintaining the game state by collecting updates from all clients, resolving inconsistencies and propagating the most current game state.

When playing over the internet, users are subjected to internet service conditions that may not always satisfy the game's QoS (Quality of Service) specification. Latency is often the greatest network related issue because it affects the usability of the games, compromising users QoE (Quality of Experience). However, for a RTS, the latency can achieve high values without affecting the game.

Maintaining game state consistency on a RTS game is a complex task. To reduce the amount of data on a state update process and still provide the user with the relevant state information, MMOGs employ *interest management* (IM) techniques[4,12]. This concept restricts the amount of data exchanged during the update process by limiting the area of interest (AoI) of a player. Thus, the player only receives the game state data necessary, mostly corresponding to the area where avatars and events relevant to him are located and take place.

This work proposes the adaptation of a continuous consistency model to the needs of the RTS games. To achieve it, we start by identifying the consistency requirements of real-time strategy and how the continuous consistency model can meet them without compromising the dynamic of the game. Later, we analyze how the concept of area-of-interest can be combined with the continuous consistency model.

2 Related Work

In the field of Distributed Systems, high availability and performance are two major requirements assured by the use of data replication. As there are multiple replicas there is also the possibility of information disparity between them, leading to inconsistency. In the next sections we will present two distinct classes of replication techniques: *pessimistic replication* and *optimistic replication*, followed by the study of three systems that seek a balance between them. Next, we explore the topic of interest management and some adopted solutions.

2.1 Pessimistic Replication

Pessimistic replication techniques aim at maintaining single-copy consistency, usually by usage of locking mechanisms[8]. These provide strong consistency by preventing conflicts between updates since data changes are isolated and propagated to all the other replicas, becoming available for all the following read operations. Despite being widely used in commercial systems[6] due to performing well in local-area networks, this solution is not suited for the Internet, since it cannot provide good performance and availability.

2.2 Optimistic Replication

Optimistic replication algorithms[9] are based on the assumption that conflicts between updates occur only rarely and can be easily fixed when they happen. As a result, optimistic algorithms allow replica data accesses without previous synchronization between all replicas. These approach introduces the concept of *eventual consistency* which states that even though the state of replicas may diverge, the continuous propagation of updates in background will eventually lead to the full consistency of the system. Since optimistic replication can support partially inconsistent data it is unsuitable for systems with strict consistency requirements that cannot tolerate occasional conflicts.

2.3 Case study Systems

Next we present a review on three divergence bounding systems, representing the evolution of the optimistic replication scenario and how it is possible to achieve a balance between data consistency and availability: The TACT framework[13], the VFC model[10,11], and finally, Data-aware Connectivity[2].

(a) The TACT framework is a continuous consistency model that enables the designer to bound the consistency requirements of the application using *conits*, i.e. units of consistency. Thus, it allows the designer to define the maximum semantic distance between replicas and even different consistency boundaries per-replica. When the operation meets the pre-established requirements it proceeds locally, otherwise it blocks the operation until synchronizing with other replicas as specified by the system's consistency requirements. TACT contributed to the consistency scenario by presenting a solution where different consistency levels can be assigned to different replicas, taking advantage of this property to increase the performance of the system by routing clients to replicas that meet their consistency requirements.

(b) The Vector-Field Consistency model is, as the TACT framework, a continuous consistency solution that allows a per-replica consistency level specification using a 3-dimensional consistency vector specifying: the maximum time a replica can be without the last update values; the maximum number of lost updates supported; and the maximum divergence between replicas or against a constant. Moreover, VFC applies the concept of *locality-awareness*, establishing *pivot points* from which is calculated the distance to the surrounding objects. A pivot point represents the position around which there are strong consistency requirements, weakening as the distance from the pivot increases. In order to define the consistency bounds, delimitating the authorized divergence degree, with the pivot as the center, there are drawn ring-shaped, concentric areas presenting increasing radius and divergence degree, called *consistency zones*. VFC proves to be an very flexible solution as it allows explicit and

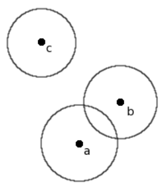

Fig. 1. *Aura-nimbus* IM

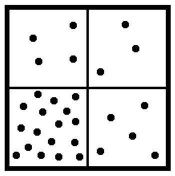
Fig. 2. Region-based IM

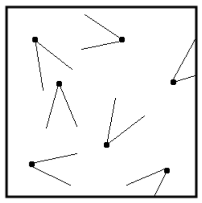
Fig. 3. Visibility-based IM

dynamic definition of consistency degrees, suitable for a wide variety of systems.

(c) Intended for the mobile environment, Data-aware Connectivity[2] rests on the principle of ensuring acceptable quality of a replicated object at minimal connectivity resources cost. To reach this goal, the system implements a continuous consistency model that enforces data consistency bounds of local replicas, along with a *connectivity monitor* that, using knowledge about the existing communication supports, calculates the less expensive connection to obtain the desired quality levels. While the previously presented consistency models, ensure replica quality by forbidding access to replicas once they exceed the allowed divergence degree and until they are obeyed again, in this system, when replica quality is below the bounds defined, it probes available connectivity in order to reestablish consistency levels, forbidding replica access only if no connection is possible.

2.4 Interest Management

In order to provide a shared sense of space among players, each player must maintain a copy of the **relevant** game state on his computer. The simplest approach requires every player broadcasting updates to all other players so that each player can maintain a full copy of the game state. However, this solution is proved to scale poorly. IM techniques[1,4] reduce the number of updates to be transferred to other players by utilizing filtering mechanisms such as application specific and/or network attribute specific. In this section we focus on application specific IM politics and classify a number of IM algorithms according to them. Interest Management schemes are usually classified, according to the range of action, as based in: *aura-nimbus*, *range* or *visibility* (Fig. 1, 2 and 3).

To obtain the finest-grained update filtering, many systems opt to apply more than one IM model. One example of this is the already referred RING system. Designed to support a large number of players in a shared environment, the RING system applies both a range-based model, limiting the AoI of every object to the region they are positioned, and a visibility-range model, further restricting the interest zone to the area the player's visibility can reach.

A3[3], an algorithm intended for the MMOGs environment, combines the aura-nimbus and the visibility-based models in order to limit the AoI of the player to an area of 180 degrees in front of the *avatar*, ignoring objects localized in the back. This algorithm also considers a second aura of smaller dimensions, retrieving information about only the closest objects behind the avatar.

VFC[10,11] applies multiple concentric aura each with different radius. This design complements the VFC consistency model, limiting *consistency zones* as rings around a *pivot* point (Fig. 4). This way objects situated in rings closer to the *pivot* point present greater consistency requirements which decline as the rings go wider. Moreover, the VFC interest management technique is also supported in a 3-dimensional environment (Fig. 5). This can be accomplished by porting the shape of the AoI to the 3D spectrum (concentric spheres). In the solution, objects situated within *consistency zones* on top of the *pivot* point present the same consistency requirements any other object in the same zone.

2.5 Global Analysis

One of the aims of MMOGs is to provide an environment where realistic interaction among players can occur. For that, every participant should be unaware of any data inconsistency regarding entities of the game.

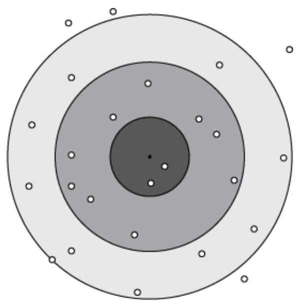

Fig. 4. 2-dimensional VFC

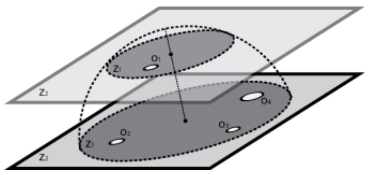

Fig. 5. 3-dimensional VFC

In order to provide a shared sense of space among players, each player must maintain a copy of the relevant game state on his computer. The simplest approach requires every player broadcasting updates to all other players so that each player can maintain a full copy of the game state. However, this solution is proved to scale poorly.

After exploring the state-of-the-art in the fields of consistency management and interest managing, it is possible to conclude that consistency maintenance is achievable at low bandwidth costs by dynamically identifying the different consistency requirements objects present at runtime.

This way, a model like VFC, which combines IM techniques with continuous consistency management, is a suitable choice for reducing bandwidth requirements and server CPU load by distributing it through the clients.

3 Architecture

In this section we start by examining the VFC model and how it can be adapted to the RTS environment, we present the architecture of the VFC-RTS middleware and how it can be applied to a RTS game. Finally, we explain in detail the consistency enforcement on the VFC-RTS middleware.

3.1 Applying VFC to Real-Time Strategy games

VFC is a continuous consistency model that employs locality-awareness techniques in order to maintain the consistency requirements of each client's local replica, without propagating the full game state.

The VFC model represents the virtual world as a N-dimensional space populated by game objects which can be player's units or structures, bonus items or computer controlled entities. The state of the virtual world is defined as the aggregation of the states of all active game objects. Each player keeps a local *replica* of the game state, however, only the server holds the most accurate global state. Responsible for update reception and conflict resolution, the server is the key enforcer of the VFC model, sending object states according to the consistency requirements established on a player by player basis.

On VFC, the user's AoI is defined using the concepts of *pivot* and *consistency zone*. A pivot point represents the position around which the local replica presents strong consistency requirements, weakening as the distance to the pivot increases. One possible adaptation of this concept for RTS is establishing a *pivot* on each object controlled by the player. However, this solution does not fill all the consistency requirements of real-time strategy since the player may not be aware of events taking place on the area captured by the camera if there are no player units present. Also, this solution may require excessive calculations as we will describe further on this document.

To enforce the boundaries between consistency levels, the VFC model introduces the concept of consistency zones. In practical terms, the consistency zones are used to quantify an object's importance. This way, objects positioned in zones closer to the pivot present more relevance and as such are refreshed more often than objects further apart from the pivot point. Since this work targets RTS games, which nowadays are most commonly 3-dimensional, we opted to carry over the original consistency zones design to 3D, describing consistency zones as concen-

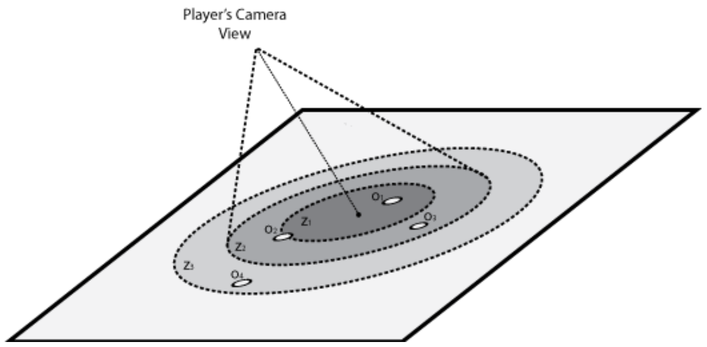

Fig. 6. Adaptation of player's AoI to his camera view

tric spheres.

The proposed adaptation of the VFC model is composed of two parts: adaptation of the player's area-of-interest to the map area captured by the camera and aggregation of entities into units according to the distance among them.

In Fig. 6 we present an illustration of the proposed solution. By assigning the pivot point to the center of the player's camera view, we guarantee that every object within the player's area of visibility presents an acceptable divergence degree for the correct execution of the game. We believe this pivot point attribution best serves the consistency requirements of real-time strategy since the majority of strategy commands and decisive battle events concerning the player take place inside his camera spectrum. Although the presented adaptation of VFC covers most of the player's area-of-interest, there can also be events affecting the user not followed by the camera. To cover this situation, we propose next an additional feature to the VFC-RTS solution - the concept of entity aggregation into units.

A unit is a group of entities controlled by the same player performing the same set of actions within a limited time period. If the system detects entities within a fixed distance threshold going on the same direction, it establishes a pivot point in the center of the unit formation and consistency zones. Not only this process allows following entities outside the camera span without compromising the consistency of the system, it also reduces the processing and communication load since update messages regarding entities of the group can be aggregated.

3.2 Middleware Architecture

Next we will present in detail how a system using VFC-RTS manages game state consistency by explaining the message exchange protocols and the different components involved in game state updating.

Following the architecture adopted by the original VFC model, the VFC-RTS distributed framework is composed of two types of network nodes: **client** nodes and a single **server** node. Running an instance of the game, clients are accountable for any changes to the game state and submission of local replica modifications to the server node. The server node is responsible for game state update reception, applying newly received updates to the global game state, solving possible conflicts between concurrent client updates and propagation of the most recent game state according to each player's consistency requirements.

Although a client-server topology is a less scalable solution when compared to a totally distributed architecture, it provides several advantages, from centralized game state control and conflict resolution, to a simplified middleware integration since the all VFC reasoning is confined to the server. Additionally, the client-server topology is the easiest and most intuitive architecture to implement, granting the additional bonus of always having a node presenting the full game state, beneficial for system monitor purposes, security control and cheating avoidance.

Client The Client node corresponds to the adapted game application run by the user. Asynchronously

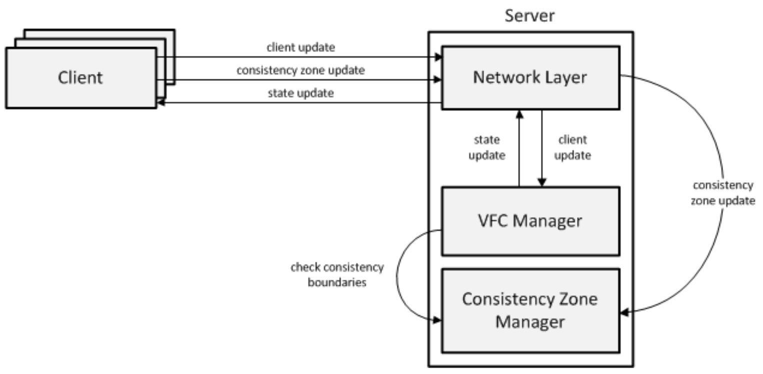

Fig. 7. Client-Server Message Flow

modifying its local replica, the client is given the illusion of interacting directly with the full game state. Whenever the client modifies the local game state, an update is sent to the server node. The client-side VFC-RTS Middleware presents a key component called **Consistency Zone Monitor**, responsible for the detection of changes to the consistency zones configurations and forwarding them to the server. Hence, whenever the player changes his camera view position (above the configured threshold value), or changes the location of a group or its configuration (i.e. number of entities in the group, max consistency radius), a new consistency zone message is dispatched to the server.

Server The Server is the central node in the system. Every message exchanged in the network is either sent or received by this node. It accumulates client management and update propagation responsibilities. Although the server immediately applies client's game state updates according to the arrival order, in a VFC powered system, it is up to the server to reason when to send object state updates in accordance to each user's AoI. To do this, the server makes use of two key components: the **Consistency Zone Manager**, holding all the information regarding every player's consistency zones; and the **VFC Manager**, responsible by enforcing the VFC algorithm in conformity with each client's consistency requirements. In addition to this task, the VFC Manager is also responsible for tagging which objects have been subject to updates and, if that is the case, to which clients have the new object state been sent.

In Fig. 7 we illustrate the message flow between the clients and the server. In this representation we can easily identify the three update processes that characterize our VFC architecture: *Client Updates*, *Consistency Zone Updates* and *State Updates*.

In a client update, the client asynchronously sends to the server the information regarding modifications to the local replica so they may be applied to the primary game state and, consequently, to the remaining client's game states. This updates have to be sent immediately after the replica update to maintain the server's game simulation complete consistency. When arriving to the server, the update is interpreted by the VFC Manager in order to determine which objects are targeted for modification and then applied to the primary game state.

Triggered whenever a client changes the configuration of a consistency zone, Consistency Zone Updates provide the server all modifications regarding the player's AoI. Once in the server, this information is stored in the Consistency Zone Manager so it can be looked up during the VFC reasoning process.

Considering object o and client c, whenever the VFC Manager detects that o is tagged as dirty and does not respect the consistency boundaries set for c, the server sends c a State Update Message, containing the o's most consistent state (the primary state). Furthermore, the server flags a state update regarding o was sent to c, preventing the server from repeatedly sending clients an object's state before it is once again modified.

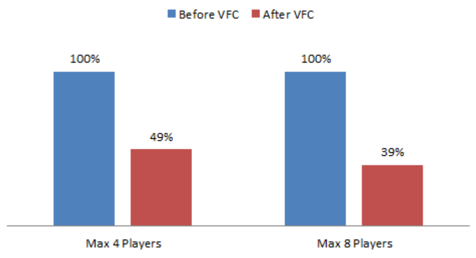

Fig. 8. Message dispatch ratio before and after VFC

3.3 VFC Enforcement

The server node is accountable for enforcing the VFC model. This particular task is assigned to the VFC Manager module. As soon as an object reaches a client's stablished VFC limits, the state of that object is immediately propagated.

To formalize the consistency degree of a zone, VFC employes a 3-dimensional consistency vector:

Time (θ): The maximum time (in seconds) a local object can stay without being refreshed with its primary replica's latest value. With this metric, VFC guarantees that an object within a consistency zone has to be refreshed within intervals of θ seconds. In the practical sense, this dimension expresses the "freshness" of an object.

Sequence (σ): The maximum number of updates to the primary object state not applied to the local object (missing updates). With this metric, VFC guarantees that an object within a consistency zone has to be refreshed within intervals of σ lost updates.

Value (ν): The maximum divergence between the contents an object's local state and its primary state. Value is an application-dependent metric and by that reason must be calculated using a function implemented by the application developers.

For the purposes of our system, ν represents the maximum distance between an object's primary replica position and local replica position, i.e., a measure of the error associated with the perceived location of an object (in the local replica) and its actual location (in the primary replica). We understand that, the bigger the consistency requirements, the smaller must the distance threshold be for replica synchronization. This feature will also guarantee the model's correct behavior in a case where an object o, presenting a great distance from any of a player p's pivots, takes a move command that will put o inside p's AoI.

4 Evaluation

Next, we present the experimental results of our evaluation of VFC-RTS. We start by presenting a quantitative evaluation consisting on a study on the use of both computational and network resources, followed by a qualitative evaluation, in which we investigate the impact of the solution to a player's perception of the game.

4.1 Quantitative Evaluation

In this section we show the results of a study conducted to the system with the objective of determining the savings in the use of network resources as well as the impact on computational performance.

Number of Messages We start our analysis by studying how applying the VFC algorithm affected server-client update propagation. Our tests were conduceted on two, three, and four players game sessions, varying for each case the size of the scenario (small, medium and large maps with a maximum capacity for two, four and eight players, respectively). The results are summarized in Fig. 8.

Analyzing to the graphic, we are able to see an overall reductions in the number of exchanged messages between 50% and 60%. Such gains are not accomplished on a two players game in the small map (for only two players) since both bases are located

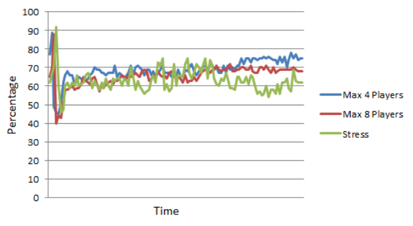

Fig. 9. CPU consumption before VFC

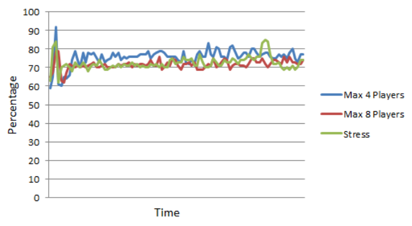

Fig. 10. CPU consumption after VFC

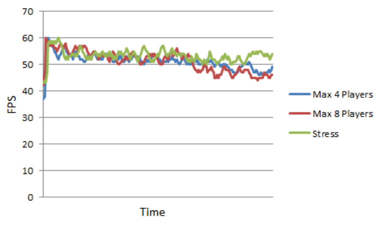

Fig. 11. Frame rate before VFC

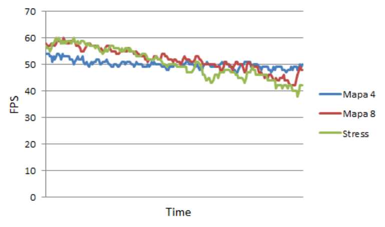

Fig. 12. Frame rate after VFC

too close to each other, resulting on an highly probable overlap between the player's areas-of-consistency. The tests also exposed how the size of the scenario has a direct impact on the update exchange rate. This factor can be explained by the increasing gap between player's headquarters as the scenario gets wider. Hence, the bigger the scenario, the more extensive is the distance between the player's AoI.

CPU Performance Next, we present the study the effects of our solution on CPU consumption. Since the majority of computational workload takes place on the server, we focus our study on this component.

In spite of being realized under the same variables as the previous test, to further examine the impact of our solution, we introduce a stress case consisting on a game session taking place on larger map where the players were enabled to make use of a game cheat that multiplies ten times the number of units controlled by him. With this test, we intend to make an assessment on how the system reacts considering a large number of game entities.

In the results (Fig. 9 and 10), we see there is an increase on the server CPU consumption in between 10 and 20 percent. This can be seen as a consequence of the additional workload required for enforcing the VFC algorithm. Despite this fact, such an increase on the server's computational requirements has little expression since CPU usage rarely surpasses 80%. Although our solution presents additional CPU usage compared to the version without VFC, the consumption rate is consistent during the entire game session, even in the stress case.

Frame Rate We proceed the quantitative analysis of our solution by monitoring the frame rate. Although this factor may sustain a direct influence on the player's game perspective, therefore also belonging in the qualitative evaluation, in this section we will solely make an analysis on how the frame frequency was affected in the system.

By looking at Figs. 11 and 12 the charts it is easily perceptible the version after shows a slight downward trend, reaching a minimal value of 37 FPS, while before VFC it presents a minimal of 44 frames per second. We believe these may come as a consequence of changes made to the update message structure in order to enforce the VFC algorithm. While, in the version before VFC, a message would simply propagate the object's latest orders, which would produce the same game state on all clients, since the system supports total consistency, the new update message structure also propagates the object's position when the order was given, so that the object may synchronize it's position with its primary replica and then proceed with the order, thus requiring additional rendering.

Although a minimal value of 37 frames per second was reached during a single stress situation, when analyzing remaining tests, we can see that the most common minimal value its about 40 frames per second, very close to the minimal frame rate before applying VFC.

4.2 Qualitative Evaluation

The main objective of this qualitative study is to assess if the introduction to the VFC model had a significant impact on user experience.

To evaluate the quality impact of our solution, we have developed a questionnaire (consult Attachments at the bottom of this document) to which every player was subjected by the end of a Warzone 2100 game session. On the questionnaire we introduce the concept of *sudden object movement*. Consisting on an object's translation between two points without describing the path in between, sudden object movements are the visual manifestation of one of two situations: *lag* over the client-server communication channel or a game state correction due to momentarily unfulfillment of the consistency requirements over the player's AoI.

The tests were conducted on two, three, four and seven players game sessions. The two, three and four players session were performed over a fixed network while the seven players games over a wireless network, which were the only game sessions where *lag* was experienced.

Upon the completion of the qualitative evaluation of VFC-RTS we could verify that:

(a) Although the latency verified during the seven players sessions may have tampered the result of this study by causing additional sudden object movements, these events caused little impact on the players strategy, hence on the outcome of the game.

(b) The game sessions conducted over a fixed network also present sudden object movements, however, the frequency of these occurrences is lower than on the game sessions over a wireless network, and presented no impact to the player's strategy or the game's outcome.

(c) VFC-RTS holds little to none overall impact to the player's game experience, however, this value is subject to the latency of the system.

5 Conclusions

In this paper we have presented a continuous consistency model which results from the adaptation of the VFC model to the RTS multiplayer environment. Besides providing a continuum between strong and weak consistency, the VFC model makes use of the notion of *locality-awareness* to define multiple zones of consistency with different consistency degrees. By doing so, the VFC model enables selective updating in accordance to a user's area-of-interest.

To start this work, we provide an overview upon the state of the art on the fields on: consistency maintenance in distributed systems and interest management techniques.

Next, we present the architecture for a system powered with the adaptation of the VFC model, describing the adjustments that, in our opinion, are required in order to best adjust the VFC model to the real-time strategy paradigm. In this adaptation, we defined the portion of the scenario captured by the player's camera as being his primary area-of-interest, therefore, the zone of consistency with the most strict consistency requirements, describing a *pivot point* at its center. Moreover, to guarantee that the player is presented at all times with the most accurate information about events regarding his troops, even if located outside the area covered by the camera, we introduce the concept of *unit*, a group of entities presenting a pivot point at its geometric center. An unit allows following entities outside the camera span without compromising the consistency of the system.

The defined architecture was implemented on top of *Warzone 2100*, an open-source RTS game with total consistency requirements. After the implementation, using a number of criteria, we analyzed the success of this work. Other than showing the correctness of the adaptation and the little impact caused to the player's game experience, the results showed the great potential of the VFC model. In comparison to the network resources usage on a total consistency system, we were able to achieve reduction by the order of fifty percent. This way, a VFC powered RTS game can maintain strict, locality based, consistency levels without compromising the player's perception of the game, an advantageous featured to games with a large number of simultaneous gamers such as MMOGs.

In this work we have established the grounds for wide variety of future works, including extensions and improvements to the current design and implementation. In this subsection we enumerate some of the lines of investigation we intend to pursue:

(a) Extend the currently centralized architecture of VFC-RTS to support a distributed server infrastructure. Since the current solution relies on a single server node, a possible improvement to the solution would be dividing the scenario into smaller areas and assigning each one to a different server. This way, we would achieve greater load balancing and also increase the scalability of the system.

(b) Enhance the consistency zones monitoring system. Currently, the VFC-RTS solution solely monitors changes to the position of the player's camera, however, modern 3-dimensional RTS games enable additional camera operations such as zoom and rotation, which we intend to take into consideration on future work.

(c) Explore the application of the VFC-RTS to a world wide global positioning system. As total consistency is impossible to accomplish on a global scale, a location aware solution such as VFC-RTS would prove to be a viable candidate.

(d) Expand the use of the VFC-RTS architecture to other online multiplayer game types that present a top-down camera perspective such as race car driving.

References

1. D. T. Ahmed and S. Shirmohammadi. A dynamic area of interest management and collaboration model for p2p mmogs. In *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, DS-RT '08, pages 27–34, Washington, DC, USA, 2008. IEEE Computer Society.
2. J. a. Barreto, J. a. Garcia, L. Veiga, and P. Ferreira. Data-aware connectivity in mobile replicated systems. In *Proceedings of the Eighth ACM International Workshop on Data Engineering for Wireless and Mobile Access*, MobiDE '09, pages 9–16, New York, NY, USA, 2009. ACM.
3. C. E. Bezerra, F. R. Cecin, and C. F. R. Geyer. A3: A novel interest management algorithm for distributed simulations of mmogs. In *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, DS-RT '08, pages 35–42, Washington, DC, USA, 2008. IEEE Computer Society.
4. J.-S. Boulanger, J. Kienzle, and C. Verbrugge. Comparing interest management algorithms for massively multiplayer games. In *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, NetGames '06, New York, NY, USA, 2006. ACM.
5. M. Claypool. The effect of latency on user performance in real-time strategy games. *Computer Networks*, 49(1):52 – 70, 2005. Networking Issue in Entertainment Computing.
6. N. Krishnakumar and A. J. Bernstein. Bounded ignorance: a technique for increasing concurrency in a replicated system. *ACM Trans. Database Syst.*, 19:586–625, December 1994.
7. J. Müller and S. GORLATCH. Rokkatan: scaling an rts game design to the massively multiplayer realm. *Comput. Entertain.*, 4, July 2006.
8. J. Munson and P. Dewan. A concurrency control framework for collaborative systems. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, CSCW '96, pages 278–287, New York, NY, USA, 1996. ACM.
9. Y. Saito and M. Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37:42–81, March 2005.
10. N. Santos, L. Veiga, and P. Ferreira. Vector-field consistency for ad-hoc gaming. In R. Cerqueira and R. Campbell, editors, *Middleware 2007*, volume 4834 of *Lecture Notes in Computer Science*, pages 80–100. Springer Berlin / Heidelberg, 2007.
11. L. Veiga, A. Negro, N. Santos, and P. Ferreira. Unifying divergence bounding and locality awareness inreplicated systems with vector-field consistency. *Journal of Internet Services and Applications*, 1:95–115, 2010. 10.1007/s13174-010-0011-x.
12. A. P. Yu and S. T. Vuong. Mopar: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In *Proceedings of the international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '05, pages 99–104, New York, NY, USA, 2005. ACM.
13. H. Yu and A. Vahdat. Design and evaluation of a continuous consistency model for replicated services. *ACM Trans. Comput. Syst.*, 20:239–282, August 2002.