First-Person Shooter for Tablets - FpsTab

Oleksandr Bodashko
olexandrbodashko@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2013

Abstract

Nowadays, mobile devices play an important role in our daily lives. Their high performance makes them not only in good working tools but also in gaming consoles. In this context, multiplayer games on *ad-hoc* networks (where does not exist any well defined structure or fixed topology) places multiple challenges. In this paper we address the following objectives: i) minimize the amount of information exchanged between players for bandwidth saving, ii) increase game scalability and iii) minimize energy consumption and resource utilization of the device. Obviously, developed solutions should not harm the gameplay. Thus, we designed and developed a middleware software called FpsTab, that offers a set of features for the aforementioned games (multiplayer games on *ad-hoc* networks) for the First-Person Shooter genre. The FpsTab offers an innovative solution that allows: i) minimizes the resources used by mobile devices (e.g. data sent/received on the network) thereby contributing to increase of system scalability, and ii) ensures game's consistency on each device. FptTab was tested by porting of the game Quake 3 Arena. Obtained results in the scope of energy consumption and resource utilization of the device, that saves bandwidth without affecting the gameplay, are promising.
Keywords: Mobile devices, Resource management, Interest management, Consistency, *ad-hoc* network, Multiplayer game

1 INTRODUCTION

Nowadays, we can't imagine our life without the personal computer. PC has revolutionized our world and became an indispensable tool for work, communication, data storage, entertainment, etc. However, with the advancement of technology, the computer had a tendency to become smaller and smaller, until now fits in the palm of the hand. Then appeared mobile devices such as smartphones and tablet PCs, that for some tasks, can replace conventional PCs.

The great performance of mobile devices's GPUs transform them into video game consoles with 2D and 3D high quality graphics. Most of them already supports OpenGL 2.0, Direct3D Mobile and programmable shaders that allows to play games with high quality textures and shadows, and enjoy a more realistic gameplay.

Between different video games genres existed for mobile devices, the genre FPS - First-Person Shooter have achieved a top of popularity, especially multiplayer FPS games.

In order to on-line game provides good performance it is necessary to considerate two factors: consistency and scalability. It's crucial maintain the game state consistent in real time, i.e., each player has a local copy of the global game state, that contains information about all players. This maintenance has a cost, when more players persist on the game more information is exchanged between them, and more bandwidth is used. There are several techniques to minimize the amount of messages exchanged in the network, such as Interest Management [14, 2] and Dead Reckoning [10].

The Interest Management consists on disseminate the game state updates only for players who are interested in this information updates. Each player has an associated area of interest (AoI). Usually this is the area around him, which is also called aura. Thus, the player receives updates only from players and entities that are within this area.

Another important factor that enables to support multiple players at the same time is scalability. To make the game scalable is necessary to choose the architecture that better fits the game. The architectures used in mobile games are: Client-Server [9] where a device, called a server, contains the entire game information that is disseminated to the players (clients); Peer-to-Peer (P2P) [7, 1] where doesn't exist definition of the central node, each node performs functions of server and client; Server Networks [9] where there are more than one server to support a higher number of connections.

However, mobile devices have one weak point -

energy consumption. When more resources of mobile device are used - more energy is spent. Energy consumption can be minimized by reducing the usage of expensive resources.

The main objective of this project is to minimize the amount of information exchanged between players, by using Interest Management techniques. For this purpose, the FpsTab defines the areas of interest (AoI), such as the field of view and the aura that significantly reduce the amount of information required to process the game. The FpsTab system performs a filtering of propagated information on the network without harming the game consistency and the gameplay.

Increase the scalability of the game is another goal of the project. The scalability depends, not only of the game server features, but also of bandwidth. Thus, bandwidth increasing allows to increase directly a scalability.

Another challenge of this project is to reduce the energy usage of the mobile device. The reduction of resource usage such as CPU, RAM and Wi-Fi helps to preserve battery life.

Find an opensource, and also FPS game, for mobile devices is the main and tricky challenge of the project. Since doesn't exists an opensource FPS games for mobile devices, we decided to port a game made for other platform. We choose Quake 3 Arena[1], because this game has an Android port client and is supported by majority of mobile devices.

Another key challenge is related to the nature of FPS games. As they are action games, it is important that to the player be aware of any events around him. We can't consider as important only those entities that are within the field of view of the player. Often, the enemy's positions are revealed by sound effects that they are producing outside the player field of view.

This document is organized as follows. Section 2 describes the related work which focuses on existing solutions to solve the problems of scalability, consistency and Interest Management; in section 3 we present the FpsTab architecture; details about system implementation are presented in section 4; the evaluation and obtained results are described in section 5; and finally, in section 6 we summarize this work with some ideas for future work.

2 RELATED WORK

2.1 Mobile games

Mobile games has evolved over the past few years, thanks to technological advances of the device components. Between different video games genres existed for mobile devices, the genre FPS - First-Person Shooter have achieved a top of popularity, especially multiplayer FPS games.

The FPS games are based on battles with weapons, those may be real or fantastic. The main characteristic of this genre is that the player can see the virtual world in the first person view, i.e., through the eyes of the main character, called avatar.

Each avatar has its associated state that is characterized by position, score, weapons and ammunition, etc. This state is changed through interaction with avatars and other entities or through their own actions.

For the consistent game it is necessary that the global game state have to be synchronized between the players. To solve this problem could be used replication technique, where each client keeps a copy of the global game state. The gameplay can also be affected by network characteristics, namely bandwidth and latency of communication.

2.2 Communication

On a *ad-hoc* network doesn't exists any infrastructure, such as access point or centralized control. Each node operates in the peer-to-peer distribution, acts as independent router and generates independent data. The topology of *ad-hoc* network can change constantly and unpredictable [5].

Network latency, the time that message takes to go from one node to another, has high impact on online gaming and can harm the gameplay. The player whose packets suffer high latency, relative to other players, has harmed gameplay, because his game state is out of date and it distorts the perception of reality in the game. The maximum latency for FPS games should not exceed 75 ms [4].

Mobile devices have various communication technologies that support ad-hoc networks. The most popular are Bluetooth, Wi-Fi and 4G. Each has its advantages and disadvantages relative to others.

2.3 Architectures

In the Client-Server architecture the information processing is divided into two distinct modules. The first module, called server, is responsible for maintaining the information and the second, called client, is responsible for obtaining the data. However, there are two approaches of this architecture where the game's logic is centered on the server or on the client (fat client) [7].

The P2P architecture is an architecture where the functions are decentralized, i.e., there is no central server, each node plays the role of client and server simultaneously [8].

2.4 Consistency

The mechanism called replication [12] is based on saving a copy of the game state on the player device.

[1] http://quake.wikia.com

The client only accesses to its local replica that is updated with the meta-information from other players. The replica content must be the closest to the overall state of the game, otherwise, the gameplay can be affected. To maintain all replicas in consistent exists replication mechanisms. They decide how often the replica must be updated, and resolve conflicts that arise when different clients change the game state at the same time.

Replicas can be managed through two approaches: pessimistic replication and optimistic replication. First approach locks access to the replicas during an update, but keeps all replicas in consistency. An optimistic replication doesn't lock the access to the replica, but the information can diverge between clients.

2.5 Interest Management

The updates that clients receive are usually generic and contain a lot of unnecessary information. The Interest Management (IM) [2] aims to filter the information, so that clients receive only what is relevant to them. The filtering of information is done by dividing the virtual world in areas of interest (AoI). Thus, avatar only receive the information for complete from his AoI.

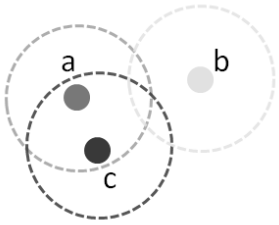

Figure 1: Example of aura based IM.

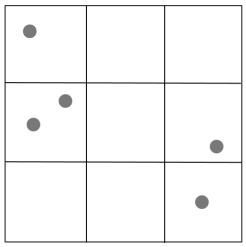

Figure 2: Example of region based IM.

In aura based IM [3] the aura is represented by circle drawn around the avatar. The area of the aura usually corresponds to the avatar sensory limit(Figure 1).

In region based IM [3] the virtual world is split into a regions. The avatar receives updates only from the region in which he is interested, generally from the region where he is located (Figure 2).

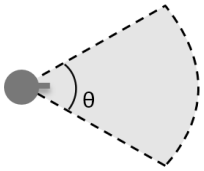

Figure 3: Example of field of view based IM, where θ represents the angle of aperture.

And last, field of view (FOV) based IM [3, 2] can filter an objects that are outside of the avatar's FOV (Figure 3).

2.6 Dead Reckoning

Dead Reckoning technique [10] reduces the amount of messages which are exchanged in the network. This approach can compute a new entity position based on its previous position, velocity, acceleration and orientation. Dead Reckoning allows to mask the latency and reduce the number of communications with the server.

2.7 ANGEL

Based on hybrid P2P architecture, keeps all devices synchronized to ensure the game consistency [6]. The master node builds a routing tree and choose the best way to pass the information to terminal nodes. The biggest disadvantage of this system is the lack of IM.

2.8 MORAP

MORAP [11] is based on the decentralized P2P architecture, where each node dynamically creates a connection with its neighbor node. This approach uses region based IM (hexagonal division) and aura based IM. MORAP is a scalable system, fault tolerant, and minimizes the amount of information in the network due to its architecture.

2.9 Adaptive Client-Server architecture

This approach [7] is based on client-server architecture. At real-time separates game logic between the server and client, based on the context and network latency. To minimize net traffic this architecture uses region based IM, aura based IM and Dead Reckoning.

2.10 Vector-Field Consistency (VFC)

VFC [13] is a mechanism to reduce the amount of information that is disseminated in the network and maintain the game state consistent without harming the gameplay. This approach introduces the concept of multiple concentric circular zones (auras) with decreasingly consistency requirements. The VFC uses an optimistic consistency model that allows replicated objects to diverge in a limited way.

The consistency level is composed by three-dimensional vectors, that defines the limits of divergence between local replicas in time, sequence and value.

This system provides consistency guarantees in the low bandwidth networks, and also is flexible and reduces network utilization.

2.11 Energy consumption

The battery life of the mobile device depends, in general, on how it's used.

Due to the large volume of graphic computation on CPU/GPU and high display quality requirement, video games have become one of the most power-consuming application. During the game with low user interaction, the total consumption of the mobile device varies between 1140 mW and 1750 mW. In the games that involves a lot of interaction, such as frequent touches on the screen and use of sensors like accelerometer or gyroscope, consumption increases up to 1640 - 2220 mW. To understand the magnitude of these values, web browsing consumes only 600 - 1500 mW.

During the game the most energy consumption component is the CPU, but other components, such as Wi-Fi, Bluetooth and display also affect energy consumption (Figure 4). By minimizing and optimizing their usage it is possible to preserve the battery life.

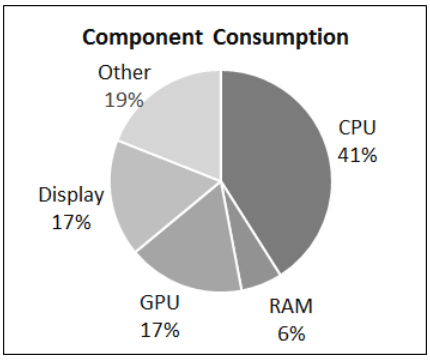

Figure 4: Average energy consumption during the game.

3 ARCHITECTURE

3.1 Overview

The fundamental idea of our system is based on the using of IM techniques. We define the areas of interest as FOV, covering the area of visual perception of the player, and aura, which defines the hearing perception.

However, we adopt the idea of the VFC system, and define various consistency levels for each AoI. This allows to reduce consistency gradually, depending on the entity position in relation to the player. Thus, we get a better trade-off between information filtering and gameplay.

3.2 Overall system architecture

The overall system architecture, as seen in Figure 5, is composed by multiple layers.

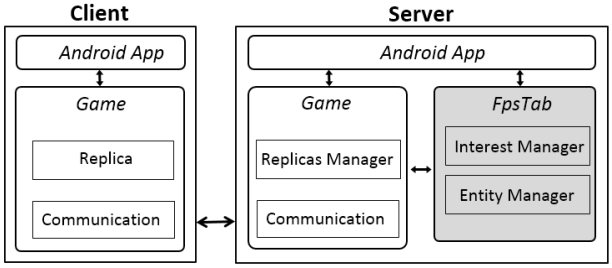

Figure 5: Overall system architecture.

The Android App layer which is located on the client and server, aims to run the game Quake 3 Arena on mobile devices with Android operating system. For that, Android App communicates directly with Game layer, that contains specific files of Quake 3 Arena. This layer also allows to switch between the original game and the game with FpsTab upgrade.

As Quake 3 Arena has the server-centric game logic, the server is responsible for game processing and managing the global state. The client only sends the inputs (avatar's actions) to the server and renders the game on the device, according to the data contained in his replica. The replica contains the avatar state (e.g. position, speed, animation) and state of other entities (e.g. opponents, items, objects).

FpsTab layer intercepts Replicas Manager and filters the information contained in the replica, before sending to the client. FpsTab uses IM based on the combinations of aura and FOV with different consistency levels. Parameters, such as angle of aperture and aura radius, can be configured trough Android App.

The communication between the server and clients is done through the Communication module located in Game layer. This module encrypts the data using a shared key, then compresses them by Huffman encoding and sends over UDP/IP protocol.

3.3 FpsTab architecture

Interest Manager module is based on IM techniques. We define two AoI associated for each player: FOV and aura. The FOV (Figure 6), similar to the human field of view, allows to see the entities that are within the visual limits of the avatar. The entities that are outside the FOV, for example in the back of the avatar, will be ignored, as they are

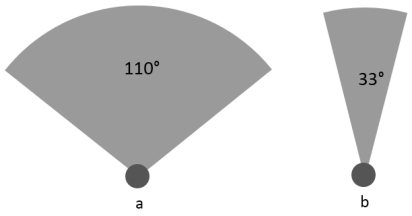

Figure 6: FOV on normal state (a) and with weapon zoom enabled (b).

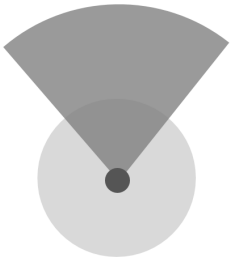

Figure 7: Combination of aura and FOV.

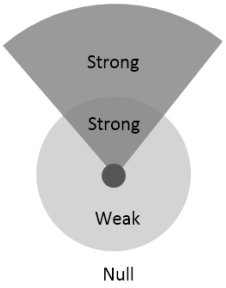

Figure 8: The consistency levels.

invisible.

We define two FOV angles: 110° for normal state and 33° when weapon zoom is turned on. The angles are slightly larger than predefined by Quake 3 Arena. This increase is made to avoid the problems of inconsistencies that may arise when the avatar performs the sudden rotational movement.

However, entities that are outside of the visual limits of avatar can not always be ignored. For example, when an avatar stays on the back of the other, his actions, such as shooting or take items, can only be captured through hearing. Moreover, the hearing perception is related to the nature of FPS games and plays an important role.

For this purpose has been introduced an aura around the avatar. As seen in Figure 7, the aura is represented by a concentric circle centered on the avatar, and its area corresponds to his hearing limit.

The level of importance of entities depends on their position and their distance from the avatar. Thus, there are three levels of consistency: Strong, Weak and Null (Figure 8). At Strong level, all entities are considered important, as they represent the highest interest for the player. The weak level of consistency, which is outside the avatar's FOV but within hearing range, considers only the sounds emitted by other entities. Finally, the Null level ignores all entities because they do not affect the player state.

Entities Manager aims to modify the state of entities that are on the replica. This allows to reduce the volume of replica, so the player receives only an information that he needs. To change entities state, this module is based on consistency levels (Strong, Weak and Null) provided by Interest Manager.

When the entity is inside the Strong consistency level, the Entities Manager doesn't change her state, contained in the replica. However, the entity state will be changed if the entity belongs to the Weak level, in order to, preserve the information to reproduce only the sound events. Finally, if the entity belongs to the Null consistency level, her state will be removed from the replica.

4 IMPLEMENTATION

4.1 Development Environment

The development of FpsTab middleware and compilation of Quake 3 Arena were made on Linux.

FpsTab was developed in a library form and implemented in *C*. Quake 3 Arena is also implemented in *C* and compiled for ARM architecture through Android NDK tool. The client, that runs the game on Android operating system (Android App), was developed in the ADT (Android Developer Tools) and implemented in *Java*. The communication between the code implemented in *C* and *Java* is made through the JNI (Java Native Interface).

4.2 FpsTab API

FPSTAB_Init Initializes FpsTab parameters.

FPSTAB_getEntityConsistencyLevel Sets entity consistency level. Receives avatar position, player orientation angles (pitch, yaw, roll), parameter that indicates whether the avatar has zoom activated, entity position and entity id.
Returns an integer that defines the level of consistency:

- 2 - Strong consistency level.
- 1 - Weak consistency level.
- 0 - Null consistency level.

FPSTAB_ShouldEntityBeRejected Tests if entity has to be removed from the replica. If function returns 0, then entity remains in the replica, if returns 1 - the entity is rejected.

FPSTAB_ManageEntityState Modifies, if necessary, the entity state which is contained

in the replica. The function returns modified or unmodified state.

FPSTAB_SetSettings Receives FpsTab configuration parameters, such as FOV angle on the normal state, FOV angle with zoom and the aura radius.

4.3 FpsTab Data structures

As FpsTab library is developed in *C*, the data structure is separated into modules (files). Each file contains functions and variables required for its operation. To facilitate integration with the library, all functions are described in a single header file (*fpstab.h*).

4.3.1 fpstab.c

FPSTAB_Init. A function that initializes the FpsTab. Resets Entities Manager temporary data.

Log. Allows to debug the code implemented in *C*. Sends a text sequence to the Android ADT.

4.3.2 fpstab_im.c

This file contains the structure of FpsTab Interest Manager. *FPSTAB_getEntityConsistencyLevel* is the main function of this module.

For consistency level calculation, the function checks if entity is inside of some avatar's AoI. The procedure is done in the order described by pseudocode:

```
if entity_is_inside_FOV then
  return STRONG_LEVEL
if entity_is_inside_aura then
  return WEAK_LEVEL
else return NULL_LEVEL
```

Function *EntityInsideFOV* defines if entity is within player's FOV and *EntityInsideAura* defines if entity is within player's aura.

The consistency level of each entity is sent to the Entities Manager through the function *addEntityToEM*. Finally, function *FPSTAB_SetSettings* defines the parameters of FpsTab.

4.3.3 fpstab_em.c

This file contains an Entity Manager structure.

The consistency level of the entities is stored in a array of integers, called *entityVec*. The vector's position matches the entity id and the value corresponds to the level of consistency.

The function *FPSTAB_ShouldEntityBeRejected* is based on the values of *entityVec* to determine if entity should remain in the replica.

FPSTAB_ManageEntityState manages the state of each entity. This function is implemented to accept the entity state, which is a structure of type *entityState_t* defined in Quake 3 Arena. When entity state is modified, all unnecessary attributes are set to 0. Thus, when the server calculates the difference (delta) between current state and the former, these attributes will be ignored. Therefore, it's possible to minimize the size of the entity state and consecutively the replica volume.

4.4 Game layer

This layer contains the game Quake 3 Arena. The game source code was obtained from open-source project called *ioQuake3*[2].

Quake 3 Arena is compiled, specifically for processors with ARM architecture, into Shared Libraries (*.so*) through the Android NDK tool.

In the original game, the server doesn't know if the player has enabled zoom. Therefore, the game was modified, i.e., the client sends a specific command to the server whenever enables or disables zoom. The server contains an array of integers (*clientVec*) that stores the state of a zoom for each player. Thus, vector position corresponds to the player's id, and the value corresponds to the zoom state (0 - disabled, and 1 - enabled).

4.4.1 Original Replicas management

We can divide the logic of the game in the following steps:

1. The client sends his inputs and internal commands to the game server.
2. The server processes the information and updates the global game state.
3. The server generates a replica from global state, specifically for each client. The replica contains the state of the avatar and state of other entities.
4. The replicas are sent to the clients.
5. The client receives his replica and renders the game on the device.

According to studies of the Quake 3 Arena replicas, it was concluded that these contain a lot of unnecessary information. This is caused by poor Interest Management that divides the virtual world into regions.

4.4.2 FpsTab Replicas management

Like Quake 3 Arena has server-centric logic, we decided to modify the replica during its development, instead of changing the contents of the completed replica. FpsTab decide whether the state of an entity should be added to the client's replica when the server fills it. Thus, we can increase the system performance, because we avoid the additional data processing.

[2] www.ioquake3.org

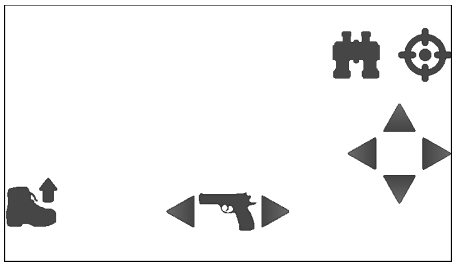

Figure 9: Virtual joystick layout.

The replicas are managed in the following order:

1. The server begins to create a replica for a particular client.
2. FpsTab is initialized (*FPSTAB_Init*).
3. While the server chooses entities to insert in the replica is called the Interest Manager, by function*FPSTAB_getEntityConsistencyLevel*, which defines the consistency level of each entity.
4. While the server adds entities to the replica, is tested whether the entity remains in the replica, by using *FPSTAB_ShouldEntityBeRejected* function.
5. If necessary, Entity Manager modifies the state of entities that remain in the replica (function *FPSTAB_ManageEntityState*).
6. The replica is sent to the client.

4.5 Android App

This layer aims to launch the game on the device with Android OS, and he is also responsible for interaction between the game and the player.

Android App was developed based on existing opensource project, called Kwaak3[3].

The original Kwaak3 executes the game on Android OS with mapping of peripherals, such as mouse and keyboard, which aren't good enough to get the complete interaction with the game on the mobile device. Therefore, we added a virtual joystick that appears on the device screen (Figure 9).

The joystick was developed in ADT, and has a graphical interface suitable for the game with the most important buttons: avatar movement, shoot, jump, zoom and weapon change. The crosshairs control is done through touch and movement of the finger on the screen.

4.6 JNI - Java Native Interface

The interaction between the Android App (implemented in *Java*) and Game/FpsTab (implemented in *C*) is done through JNI. This communication is performed at runtime and works by calling remote functions.

[3]http://code.google.com/p/kwaak3/

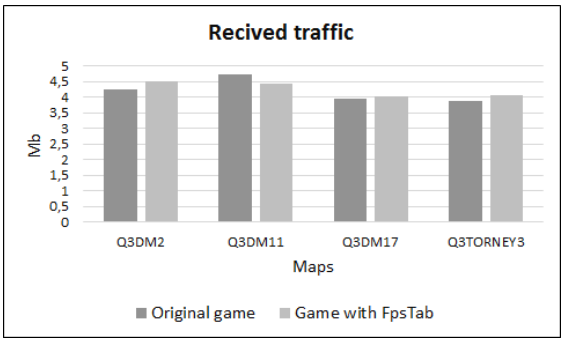

Figure 10: Traffic received by server on different game maps.

Thus, we can use functions of Game or FpsTab from the Android App and vice versa. This communication is done through the intermediary library *libkwaakjni.so*.

5 EVALUATION

The evaluation consists of comparing the modified game by FpsTab with the original game. Mainly, we had analyzed the amount of traffic produced by both solutions. The idea was to compare the resource consumption of mobile devices such as CPU, RAM and battery. However, we evaluate the quality of FpsTab to understand if the gameplay wasn't affected.

5.1 Maps

To perform the tests, we chose 4 maps with different sizes and occlusion. The maps are:

- *Q3DM2* - small and closed.
- *Q3DM11* - large and closed.
- *Q3TOURNEY3* - small and opened.
- *Q3DM17* - large and opened.

5.2 Evaluation methodology

The tests were made on four mobile devices with four players at a time. The competition mode was *Deathmatch*. Before testing all players had 5 minutes training. After training, two games were made for each map, i.e., one game for each solution (original and FpsTab). After each game we had watched the performance results and after each map, players has answered a questionnaire.

Performance measures of the devices were obtained by using System Monitor Lite[4] application.

5.3 Quantitative evaluation

5.3.1 Traffic analysis

The graph of Figure 10 shows the average amount of information received by the game server. As we

[4]http://cgollner.x10.mx

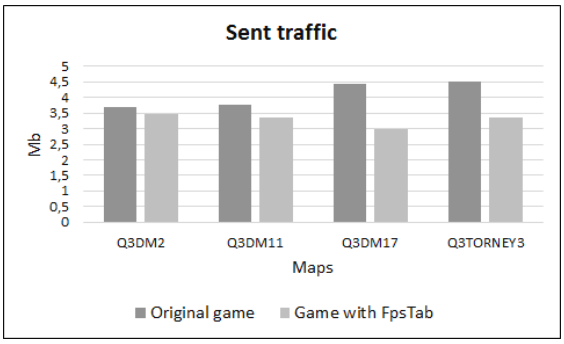

Figure 11: Traffic sent by server on different game maps.

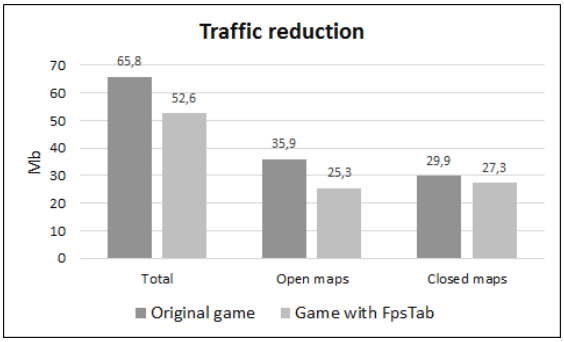

Figure 12: Traffic comparison by map type.

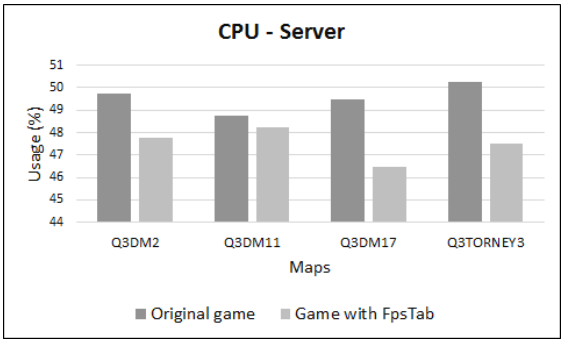

Figure 13: CPU usage by server. Comparison by map type.

can see, the traffic received by two solutions is practically the same, because he is produced by clients and is independent from solution.

However, server sent traffic depends of his management logic. As we can see in the graph of Figure 11, the server with FpsTab sends less information compared to the original game server.

In 32 games the amount of the sent data corresponds to 118,4 Mb, where 65,8 Mb has been sent by original game and 52,6 Mb by FpsTab. Therefore, the solution with FpsTab was able to reduce sent traffic in 20% compared to the original game.

The largest reduction in traffic (80%) occurs in opened maps, where the Quake 3 Arena practically don't uses its IM based on regions (Figure 12).

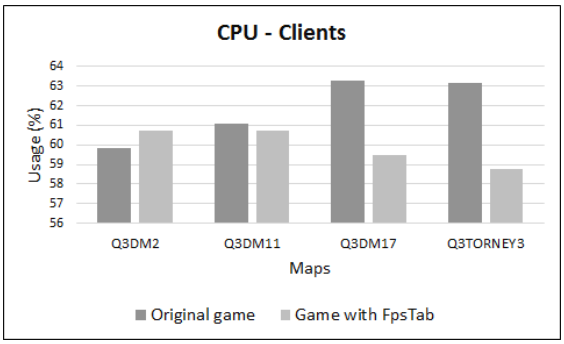

Figure 14: CPU usage by clients. Comparison by map type.

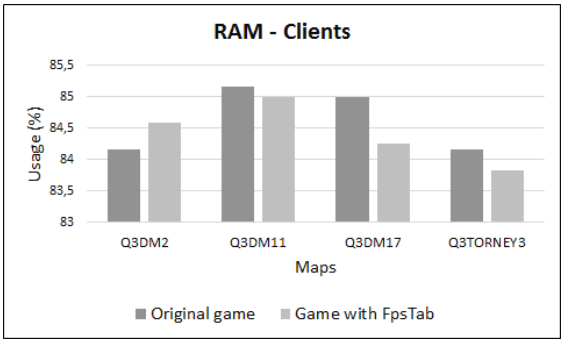

Figure 15: RAM usage by clients. Comparison by map type.

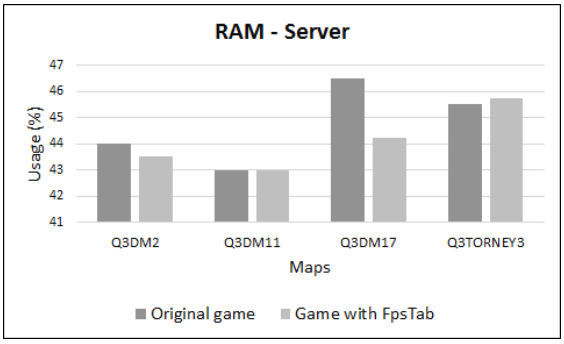

Figure 16: Ram usage by server. Comparison by map type.

5.3.2 CPU

In terms of CPU usage, the game with FpsTab has reduced processor usage by 4,2% on the server (Figure 13), and by 3% on the client side (Figure 14). Mainly, the reduction had happened due to decreasing of the amount of processed entities. The greatest CPU reduction was achieved on the opened maps, such as *Q3TOURNEY3* and *Q3DM17*.

5.3.3 RAM

As shown in the graph of Figure 15, the FpsTab reduced RAM consumption on the client by 1,4% over the original game. The largest reduction occurs in the server and contained 0,25% (Figure 16).

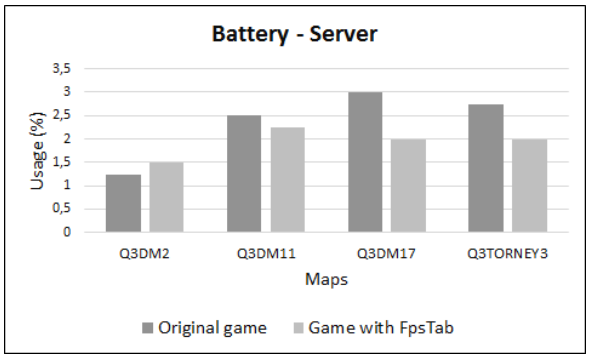

Figure 17: Battery usage by server. Comparison by map type.

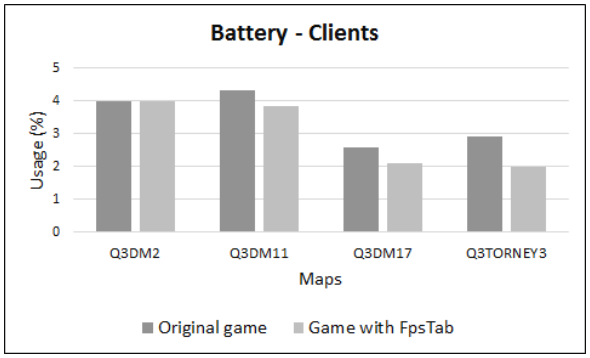

Figure 18: Battery usage by clients. Comparison by map type.

Again, we obtained better results on large and opened map *Q3DM17* with a lot of objects. The reducing on this map reaches 5% on the server and 1% on the client, in comparing to the original game.

5.3.4 Battery

The game with FpsTab spends on average less then 18,4% of battery on the server, and less then 13,9% on the client, in comparing to the original game (Figures 17 and 18). The greatest battery saving we obtained at the opened maps, as on the server and as on the client.

5.4 Qualitative evaluation

In order to evaluate if FpsTab didn't harm the gameplay, we had compared the original Quake 3 Arena with Quake 3 Arena improved by FpsTab. As the gameplay is impossible to evaluate objectively, it's measured according to the subjective opinion of the players.

After every two games, i.e., after each map, the players were interrogated to answer the questionnaire. The main question was if they had noticed the differences between the two versions, mainly related to the opponent's movement and realism of the game.

Tests were performed by 16 players on total. Only 2 of them had noticed the differences between

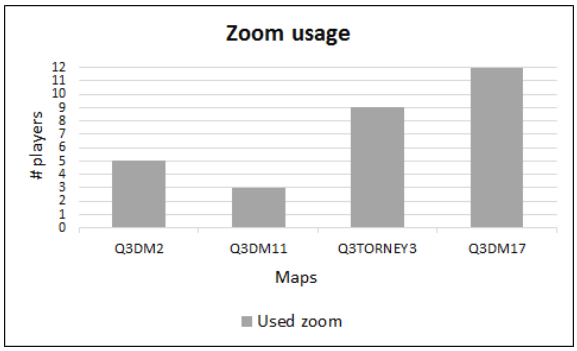

Figure 19: Zoom usage by players. Comparison by map type.

the two versions. One of the players had noticed, that sometimes entities appear with a slight delay, if he quickly switch between the normal mode and zoom and also at the same time make a sudden movement. The other player, after disabling zoom still got his FOV reduced for 1 or 2 seconds.

As the Figure 19 demonstrates, most of players has used a zoom on the opened maps (*Q3TORNEY3* and *Q3DM17*). This type of maps requires a better aim, since the entities may move on a big distances.

6 CONCLUSIONS

In this paper we have talked about the multiplayer FPS games, and challenges that they provide in terms of consistency and scalability. We also have talked about the architecture of multiplayer games, Interest Management and replication. Overall, we have mentioned *ad-hoc* networks, about the challenges that they pose, and about the main communication protocols.

We have proposed the FpsTab, which is a library to improve the Interest Management of FPS games. Our system uses IM techniques based on FOV and aura, which represent the sensory limits of avatars. This allows filtering out the unnecessary information for the player.

We prove that the FpsTab is efficient, because it can reduce the resource usage of mobile device, such as CPU and RAM. Due to its algorithm, FpsTab can reduce the computational load that benefits the client and the server. The other advantage is the reduction of the bandwidth. Our solution can reduce the size of the clients replicas, disseminated on the network, without affecting the gameplay.

6.1 Future work

Despite the good results, the FpsTab can still be improved. The main points to be improved are:

- The current system is very focused on Quake 3 Arena functioning. To make the system completely generic, i.e., to be integrated with any

FPS game, the library will still need to be improved.

- The FpsTab was tested in the game with client-server architecture, and with the logic centered on the server. Also its could be tested on the game with the same architecture, but with the logic centered on the client. Integration with P2P architecture is also something that could be explored in the future.
- To increase the performance, further can be added more IM technics and improve already existing, for example, reduce the FOV distance when it's covered by obstacles.

REFERENCES

[1] O. O. Abiona, A. I. Oluwaranti, T. Anjali, C. E. Onime, E. O. Popoola, G. A. Aderounmu, A. O. Oluwatope, and L. O. Kehinde. Architectural model for wireless peer-to-peer (wp2p) file sharing for ubiquitous mobile devices. In *2009 IEEE International Conference on Electro/Information Technology, EIT 2009, Windsor, Ontario, Canada, June 7-9, 2009*, pages 35–39. IEEE, 2009.

[2] J. Boulanger. *Interest Management for Massively Multiplayer Games*. Canadian theses. McGill University (Canada), 2006.

[3] 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.

[4] D. H. Brandt. Accelerating online gaming: Defining and defeating lag in online gaming. In *New Technology T-611, Computer Science, Reykjavk University*, 2007.

[5] I. Chlamtac, M. Conti, and J. J.-N. Liu. Mobile ad hoc networking: imperatives and challenges. In *Ad Hoc Networks*, number 1, pages 13–64, 2003.

[6] Y. Kaneday, M. Minematsuz, M. Saitoz, H. Aidaz, and H. Tokuday. Angel: A hierarchical state synchronization middleware for mobile ad-hoc group gaming. In *Proceedings of the 1st workshop on Network and system support for games*.

[7] A. M. Khan, I. Arsov, M. Preda, S. Chabridon, and A. Beugnard. Adaptable client-server architecture for mobile multiplayer games. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, pages 11:1–11:7, ICST, Brussels, Belgium, Belgium, 2010. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[8] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games. In *IEEE INFOCOM*, 2004.

[9] S. K. Opoku. A simultaneous-movement mobile multiplayer game design based on adaptive background partitioning technique. volume abs/1209.3052, 2012.

[10] L. Pantel and L. C. Wolf. On the suitability of dead reckoning schemes for games. In *Proceedings of the 1st workshop on Network and system support for games*, NetGames '02, pages 79–84, New York, NY, USA, 2002. ACM.

[11] Y. A. (Peiqun) and V. S. T. 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*, pages 99–104, New York, NY, USA, 2005. ACM.

[12] Y. Saito and M. Shapiro. Optimistic replication. volume 37, pages 42–81, New York, NY, USA, Mar. 2005. ACM.

[13] N. Santos, L. Veiga, and P. Ferreira. Vector-field consistency for ad-hoc gaming. In *ACM/IFIP/Usenix International Middleware Conference (Middleware 2007)*, Lecture Notes in Computer Science. Springer, September 2007.

[14] J. Smed, T. Kaukoranta, and H. Hakonen. *A Review on Networking and Multiplayer Computer Games*. 2002.