Abstract—With the recent evolution of network-based multi-player games and the increasing popularity of online games demanding strict real-time interaction among players – like First Person Shooter (FPS) –, game providers face the problem to correlate network conditions with quality of gaming experience. This paper addresses the problem of the estimation gameplay quality during real-time games; in particular, we focus on FPS ones. Current literature usually considers end-to-end delay as the only important parameter and deducts system performance indexes from graphical ones. Player satisfaction, on the other hand, is usually evaluated in a subjective way: asking the player, or measuring how long he/she stays connected. In this paper we use a testbed with synthetic players (bots) to directly correlate network end-to-end delay and jitter with expected players’ satisfaction. Running extensive experiments we argue about effective in-game performances degradation of penalized players. Performances are measured in terms of score and number of actions – kills, actually – performed per minute.

Index Terms—Network performance evaluation, objective evaluation, networked games, first person shooters.

I. INTRODUCTION

Over the past fifteen years the video games market has grown at an exceptional rate. This impressive progression have been sustained by the innovation and creativity typical in this kind of industry. Indeed, the continuous hardware and software technological improvements have offered to game designers new opportunities to create games based on unrelenting innovative concepts. Among the basic ingredients in recent successful games, interactivity supported by network communication is a key element. As a matter of fact, multiplayer gaming is becoming extremely popular within the video gamer’s community and gets integrated in an increasing number of products, sometime as a core element like in [1]. Under the lenses of this perspective, the possibility to predict quality of users’ experience from network conditions will be a strategic advantage for any network-based game provider.

Since broadband Internet access is now widely available to many households, the majority of players can connect with very high speed to remote servers and other players. Despite the fact a majority of players are considering bandwidth availability as a major factor and claim to be globally satisfied by the network communication service the typical bandwidth required by most of multiplayer games, such as First Person Shooters (FPS) and Massive Multiplayer Online Games (MMOG), is very low: a few tents of Kbps. As a matter of fact, the delay introduced by network transmission proved to contribute the major impact on the game play. Unfortunately, up to now and to the best of our knowledge, a limited number of contributions can be found regarding the impact of network delay and jitter on game playability and user experience. Interestingly enough, almost anyone of these contributions adopt a subjective experimental approach. The quality of gaming experience is usually evaluated asking the player how he/she enjoyed the experience, or measuring how long his/her has been connected; results are then correlated to the observed network parameters.

In order to achieve a better understating of the real correlation between network delay and players’ expected performance we propose, in this paper, an objective assessment methodology to investigate the impact of end-to-end delay to gameplay. We concentrate our study on FPS games since they are both very delay sensitive and extremely popular video games. Our methodology is based on providing a testbed platform. This platform is composed by a number of clients and one server where each client runs an autonomous player (“robots”, or “bots” for short) with very controlled and uniform skills. Using this testbed we have been able to run multiple and replicable experiments where different network conditions can be taken into account while providing indexes related to player’s in-game performance. Experimental results from this testbed show a clear correlation between delay and players’ performance in terms of fairness, score, and ranking. These results will be useful for a game provider to perform preemptive traffic engineering, evaluation of network infrastructures, and true-skill matching between players taking network handicap into account.

The remainder of this paper is organized as follows: in the next section we review related works on assessment of the gameplay. In Section III a brief outline of engines architecture for FPS games is presented. Section IV gives a detailed description of our testbed platform and experimentation scenarios. Obtained results of the objective assessments

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.
of delay impact are discussed in Section V. Finally, Section VI summarizes the main contributions of this work and provide some future research directions.

II. RELATED WORKS

Many contributions in literature start by analyzing the correlation between the evolution of gamers population on a public server and network conditions experienced by the users. In [2] players from an Half-Life server have been monitored leading to effective evidence that delay plays a critical role to keep players in the game. In [3], and later in [4], players in a Quake III environment have been considered; in both cases Internet quake servers were monitored and authors have been able to draw some considerations about delay and jitter based on user’s behavior. In the former work only delay information is considered; the general conclusion is that the lower the observed delay the longer a player will stay connected, at the same time the number of frags – i.e., temporary kills, using a military slang – does not seem to be very significant. In the latter contribution, [4], instead, jitter is also considered; authors come to the conclusion that for the game to be considered playable by users jitter should be limited to one fifth of the delay. Moreover, from the reports, there is also evidence that frag rate is related to jitter level.

Due to the extremely dynamic activity of on-line games and the short life of gaming sessions it is generally difficult to get accurate measurements. With the aforementioned methodology it is hard to understand bounds for delay and jitter because there is no reliable way to get feedback from the players on their reasons for leaving the game. For this reason in other cases experiments have been performed using a smaller population and later questioning the players.

In [5] and [6] experiments have been carried out using a lab testbed with controlled network conditions and using real players. In both cases the analysis was performed using Unreal Tournament 2003 and network delay has been artificially simulated. In [5] the gameplay is assessed by a survey submitted from the players. From this survey, authors try to figure out if a player understand when he/she have been penalized or not. Results pointed out that a delay greater than 60 milliseconds is disturbing for the user and jitter does not play a prominent role. A comparison between the subjective survey results and an analysis of game logs leaded authors to divide the players into two groups: “complainers” and “optimists”, which perceive the game quality differently and provide the general bias of this approach. In [6], instead, players have been able to notice latencies as low as 75 milliseconds and found gameplay less enjoyable at latencies over 100 milliseconds. However, the delay did no seem to effect the game scores in the tests. From analysis of the game logs authors suggested to keep the delay below 150 milliseconds to have a playable match.

Another contribution in [7] focus mainly on perceptual quality. A number of different games have been proposed to real players and feedbacks collected by means of a survey. Authors come to the consideration that players tend to underestimate their maximum tolerance to network latency, being a little too picky. Moreover, delay and jitter effect differently the proposed games; this is supposedly due to specific lag-compensation algorithms; Unreal Tournament 2004 proved to have the strongest correlation between subjective perception and the objective outcome of the game.

In [8], Quake 4 is used to develop a end-to-end quality measurement method which has been validated against subjective experiments. By applying this method, based on delay and jitter values, authors report that the Mean Opinion Score shows a very high correlation with subjective data.

The existing contributions having the closest correlation with the paper presented here are [9] and [10]. In the former work the authors setup a testbed where artificial players are used to evaluate the correctness of a proposed tool to rebalance delay and increase fairness between users. To the best of our knowledge this has been the first attempt to use synthetic players to assess gameplay. In this case the experiments have been limited to Quake II due to the restricted availability of client-side bots at that time. Moreover, despite the similar approach, the aim of the paper was to evaluate effectiveness of the delay compensation tool and fairness evaluation have been limited to understand if and how human players behavior could be correlated with bots’ behavior.

In the very recent work [10], the authors show through experiments how network latencies can affect AI agents that are totally offloaded from the game server. A third-person shooter tank is considered. In such game gender possible tank trajectories in a short interval of time can be predicted with reasonable errors. Therefore, to compensate the latencies effects the authors propose to consider a classical technique. Namely, dead reckoning. The expected advantage of this technique was quantified in [10] through two metrics: hit rate and proportion of wins. Despite the similarity of using client-side bots, the aim of the paper was to prove the benefit of dead reckoning. Moreover, the experiments were only limited to two AI agents playing against each others.

III. GAME ENGINES ARCHITECTURE

A game engine is a software providing all the functionalities needed to play video games using an hardware terminal (e.g., a console or a PC). It includes several components, the most important ones are:

- The 2D/3D graphic render is responsible of all visual game rendering such as avatars and background details. The majority of state-of-the-art graphics engines adopt an object-based representation of the virtual game environment. This representation provides simplified game design and can be used efficiently for large virtual worlds.
- The physics simulation engine takes in charge the visual coherence of avatars and objects movements within the game. Its basic functionality is to simulate the effects of physical behavior of objects, like collisions.
- The Artificial Intelligence (AI) module aims to provide rich and realistic interactions between players and environment. More precisely, it produces the illusion of natural/intelligent behavior of Non-Playing Characters.
(NPCs) simulating actions and movements as a human-controlled avatars would perform.

- The network module is of particular importance when the multiplayer mode is proposed. Basically, the networking subsystem is in charge of transmitting game’s data between all entities involved in the game.

Nowadays, most of the online multiplayer games are using a centralized client-server architecture. In this case, one part of the game software is located in each client, and the other part in a central server, which acts as a coordinator. With this architecture the gameplay is managed by having continuous communication between each user’s client and the server. More precisely, the clients are responsible of collecting and sending to the server all primitive commands related to the players actions within the game, such as avatar control commands (i.e., movements and shooting). On the other side, the server is responsible for computing the game actions received from each client. This new game state is then shared back, by the server, to all the clients. Because of the centralized nature of the architecture, the server can easily keep a consistent game state. Beside the obvious problem of scalability, the most important requirement to achieve good performances is to maintain a low round trip delay between clients and the game server. If the sum of round trip delay over the network and computational time on the server is greater than the state update interval on a client the player will suffer from poor performance. When subject to excessive delay, the player’s vision of the virtual world will not be in synch with the server and actions will be misplaced in the next state computation.

IV. EXPERIMENTAL SETTINGS

The testbed is composed of five PCs: four clients connected through a switch to a dedicated game server. The clients are. All the clients have the same hardware configuration: a 2 GHz Core 2 Duo processor, two gigabytes of main memory, and consumer-level graphic and network cards integrated into the motherboards. The server has a 2.4 GHz Core 2 Duo processor with two gigabytes of main memory. All five PCs run the same operating system: Linux Ubuntu 7.10 distribution with kernel version 2.6.22.14. We use the linux Traffic Control subsystem to emulate network delay and jitter. In this case, there is no cross traffic between clients and the server.

As FPS game we adopted Quake III. Quake III is a well known online multiplayer FPS which is, nowadays, still largely played over the Internet. Moreover, this game has been released to the open-source as iQQuake III [11].

In order to run completely objective experiments and to be able to reproduce the experiments we used autonomous bots to reproduce the behavior of each player. The kind of bot we adopted is a client-side one: differently from the classical bots provided by the Quake III engine it is run on a client and mimic a real-user behavior. Parameters of the bots have been set in a way to behave as a “average good player” and we applied the same bot configuration on all clients.

In all the experiments the same map has been used and the game mode has been set to “death match”; which means, the first player who manages to reach a certain score, by killing other players, wins the match. In each experiment one hundred consecutive matches have been played. The duration of a match varies usually between eight and nine minutes; which means approximatively fifteen hours for each experiment.

In order to avoid learning effects in the bots’ behavior we decided to kill all client processes and start them again on every match. This have been done because we observed an unfair behavior in score distribution when performing very long experiments. Results after adopting this strategy showed to be more fair, even on the long run. To achieve client restart we developed a client-server system to monitor each match: client agents receive messages from the server and interact with the quake client to obtain match synchronization and perform software restart. In detail, every experiments takes place in the following five steps:

1) the server send a START message to each client to connect into the game,
2) the game is played until one client reach a score of 40,
3) the server send a KILL message to each client to disconnect from server and kill the quake client,
4) the server eventually send a CHANGE FILTER message if any client have to modify its emulated network parameters,
5) finally, the server send another START message, and all quake clients will restart and connect to the game server.

Experiments have been run with penalizing a single player all possible combinations of network parameters: end-to-end delay, jitter and packet loss. Delay have been changed between 0, 25, 50, 75, and 100 ms while jitter values have been 0, 10, and 20 ms. Regarding packet loss, a uniform loss probability of 0, 10, 20, and 30 % have been applied.

V. RESULTS

A. Metric Validation

Score at a match end has always been considered a good metric to understand if a player is well placed in the arena or, more in general, if he/she is playing in a satisfactory way. Figure 1 outlines the score CCDF when no player is subject to any delay; as can it be observed, the match is fair because there is an extreme high probability for each player to score at least 25 but there is small evidence of an effective balance of skills in the range from 30 to 40 due to visible oscillations. These oscillations are due to the fact that there is only one winner for a match – the one reaching score 40 –; the other players are left behind by one or two points, tagging them as losers even if the quality of their match have been the same as the winner’s one.

Penalizing one of the players brings the obvious result as presented in Fig. 2. Despite the fact commercial FPSs claim to be playable with up to 150 ms delay on the round-trip we can clearly observe a performance degradation for one player with just 25 ms (50 ms round-trip delay). In a real-life environment this situation could be mistakenly classified as “fair” based on the fact that the probability for a player
to score at least 25 is still very high despite his/her changes to win have actually decreased. An interesting consideration comes from the observation that we are not actually able to detect small changes in behavior: a small penalization may get confused with the aforementioned oscillations and ignored. This is not really a problem in the current scenario but it might be in other games, depending by the performed activities.

Performance degradation based on delay seems to follow an exponential behavior, as depicted in Fig. 3. Starting from 50 ms delay – which is close to the threshold where an FPS starts to be less playable – we can actually see a significative decrease for the player to reach score 25; hence, a human player will actually start to complain.

During experiment we found out that rate of actions – or rate of frags, in this specific case – can also be used very efficiently to estimate a user’s satisfaction during online gaming. In some cases, as we are going to show, this metric gave also a better understanding of each player’s status. The equivalent of Fig. 1 and 2 based on the CCDF of inter-time between frags are respectively Fig. 4 and Fig. 5.

Comparing these results we can observe a much steadier behavior than before; making it possible to detect small penalties. Moreover, frags rate can be computed in real time: it is no longer mandatory to wait for the game to end, and we can detect a player not having fun in the middle of the match.

B. Effects of Delay

Figure 6 is the equivalent of Fig. 3 using inter-frag time distribution. This graph is confirming previous considerations about payer’s performance: degradation with delay is exponential, and 50 ms delay gives a serious handicap. In particular, during the experiments, the probability to wait more than 20 seconds to hit a player increases by 15% with 50 ms, 30% with 75 ms, and 40% with 100 ms delays.

C. Effects of Jitter

Penalty caused by jitter seems to be negligible compared to delay. In Fig. 7 the variation of frag rate is reported for jitter.
changing from 0 to 20 ms on a connection already subject to 100 ms delay. Curves describing different jitters are oscillating around the basic behavior and seem not to be a real handicap for the player. This behavior is confirming results from [4].

\[ \text{Fig. 6. Fragging performance degradation based on player’s delay} \]

D. Effects of Packet Losses

Figure 8 describes how a player’s performance is affected by packet loss on a connection affected also by a 50 ms delay. The behavior here follows again a negative exponential law, making it possible to compare (uniform) packet loss rate with (constant) delay values. In particular, observations from tested results seem to indicate in this environment a packet loss rate of 30% is very similar to increasing the delay of 25 ms without any packet loss.

\[ \text{Fig. 7. Fragging performance degradation based on jitter with 100 ms delay} \]

VI. CONCLUSIONS AND FUTURE WORK

This paper addressed the issue of estimating objectively the gameplay of time-constrained networked games. The evaluation was conducted experimentally using an FPS game setup running autonomously and simulating the realistic behavior of players, over an emulated network. The network conditions were controlled to evaluate the effects of delays, jitter and packet losses on game playability. The first contribution of this work is the identification of the score CCDF as an objective metric to the gameplay experience. The experimental results show clearly how the score CCDF can capture the fairness a party is experiencing. In addition, the experiments have also derived interesting characteristics of other metrics: e.g., the exponential distribution nature of the inter-fragging times. Finally, the results have proven the important sensitivity of the gameplay (especially regarding the fairness among players) of the delays over the jitter or packet losses. While many works in literature have pointed out that FPS games can be playable with up to 150 ms round trip delay, our experiments have shown that the game may be sensibly unfair for a player experiencing just 50 ms delay. Our future prospects aims to address this last issue and to investigate how efficiently modern lag compensation techniques can cope with these small delays.

REFERENCES