Jose Saldana, Mirko Suznjevic, "QoE and Latency Issues in Networked Games," Handbook of Digital Games and Entertainment Technologies, Published online 22 August 2015, Springer. doi 10.1007/978-981-4560-52-8_23-1. Authors' version

QoE and Latency Issues in Networked Games

Author Information

Jose Saldana

Aragon Institute of Engineering Research (I3A)

EINA, University of Zaragoza

Ada Byron Building, D. 2.05. 50018 Zaragoza, Spain

Phone: +34 976 76 2698

jsaldana@unizar.es

Mirko Suznjevic

Faculty of Electrical Engineering and Computing, University of Zagreb

Unska 3, HR-10000 Zagreb, Croatia

Phone: +385 1 6129 755

mirko.suznjevic@fer.hr

Abstract

The rise of the Internet opened new possibilities for computer games, allowing real-time interaction between players in different parts of the world. Online games permit a number of people to compete in a shared virtual world. However, the synchronization and the maintaining of a coherent game state to be shared by the applications of all the players is not a trivial problem: different sources of latency appear and may cause inconsistencies between the game states observed by each of the players. Different genres of online games present specific latency requirements, depending on the game dynamics, its characteristics, and the level of interaction between the players. This chapter discusses the different mechanisms that companies use in order to overcome the problem of network latency when providing online games: the use of low-bandwidth traffic flows, the different protocols used at Transport level, the architectures employed, the distribution of the hardware resources, the mechanisms for hiding the effect of the network to the players, etc. In addition, the different techniques used for estimating the user's Quality of Experience from network parameters are surveyed. Although latency is the most important parameter, other ones as packet loss, delay variation (jitter) or bandwidth are also considered. Different QoE-enhancing mechanisms, as client-side prediction or server delay compensation, are summarized. Other scalability-related techniques are also explained.

Introduction

The popularity of computer games is growing, and nowadays gaming industry is a solid and well established one. Although some people still think that computer games are just toys for kids, this is no longer true: according to the Entertainment Software Association (ESA) *"Top Ten Industry Facts 2013"* report, the average age of a game player in the USA is 30, and he/she has been playing games for 13 years.

There are many reasons and motives why people play digital games. Research has confirmed the impact of several factors such as challenge, freedom to act in a virtual word, opportunities to socialize with other people, etc. (Deci and Ryan, 1985). Regarding the challenging aspects, computer games try to find the equilibrium between two extremes: boredom, which is caused by trivial and non-challenging tasks, and anxiety which presents itself when the task is perceived as too hard or impossible. In such equilibrium users can be fully immersed in the game and enter the "state of flow" (Chen, 2007). Developers use different methods for making the game engaging, like the creation of immersive, complex and important stories (e.g. saving the world), high scores, records of completing certain activities in the game (achievements), etc. Some other components have a social dimension such as enabling multiplayer interactions with other players, competitiveness tools (e.g. ranking lists), etc. These social components are certainly essential: if you "save the world" but no one notices it in the real world, the reward is perceived as significantly lower than in the case you can share your accomplishments with someone else. Similarly, showing off the new powers and weapons of your virtual avatar to other players can be considered as one of the motivational aspects of the game.

As a consequence, more and more people no longer want to play *against the machine*, but they want to fight with and against real people (and beat them), because it is more challenging and rewarding. And if you defeat in a virtual world someone you know in the real world, it is even better! This effect can be seen even on mobile devices, where the majority of the games produced are simple ones intended for one player and offline play, but the games producing most revenue are multiplayer ones.

In the first years of computer games, player vs player competition was only possible in a very limited way: many games and consoles included two (or more) controllers, thus allowing real-time competition between players. It was certainly very limited: the players had to be in the same place and their number was reduced (two or four). Nevertheless, the experience was better than that of turn-based games, where only one player could use the controller at a certain moment.

However, the rise of the Internet opened new possibilities, since it allowed the interaction between computers in different parts of the world. In principle, the Internet was not designed as a real-time network, and it was not able to guarantee an upper bound for packet delivery delay. In fact, the first widely deployed services (e.g. e-mail, file transfer, virtual terminal connection) tolerated some amount of delay. However, in the last decades, as more and more households were connected to the Internet (with e.g. 14 or 33 kilobits per second modems), the Internet has been utilized for supporting real-time services as Voice over IP (VoIP) or video conferencing, where delay is critical. In this context, multiplayer online games became popular worldwide for the first time during the 1990s.

Different limits for the delay of real-time services have been found: for example, in 1996, the International Telecommunications Union (ITU) reported in their Recommendation G.114, *"One-way transmission time"* that a one-way delay of 150 milliseconds could be considered as the limit for a comfortable voice conversation. As the delay grows, the probability of the two call participants interrupting each other becomes higher.

The terms "latency" and "delay" are normally used interchangeably. In the hardware business, in general, "latency" basically means "inherent delay" (i.e. the delay caused by the underlying technology). By contrast, "delays" are due to other hold ups (e.g. packet processing, queuing due to network congestion, or retransmission of data due to packet loss). In networking terms, such delays can be incurred on every router on the network path being taken by the data. In this chapter we will use the term "latency" meaning all the time required by the information for traveling from one application layer to another.

When a game developer is designing a non-online title for a console, everything is under his control: the hardware, the software developing tools and the methodology are well-known. So their duty ends when the game box is sold to the player. However, when a game includes an online mode, the company is also in charge of the network support, and they have to maintain an infrastructure including a number of servers and network resources in order to let the game work. So selling the game box is just a new beginning. In addition, a new problem appears: although the software and the hardware where it runs are well known, there is something between the different devices that is not under the company's control: the network.

Companies employ different techniques in order to provide games in a scalable, reliable and profitable way, with the main objective of providing a good Quality of Experience. This concept not only includes Quality of Service, which is directly related to the parameters that can be measured from the network. The concept of Quality of Experience also includes the expectations, feelings, perceptions, cognition and satisfaction of the user (Rehman-Laghari et al., 2011). This means that two players may feel very different when their game is affected by the same latency: the most experienced player will feel the problem as more severe than the novel one (Suznjevic et al., 2013).

This chapter surveys the most extended practices used by game providers when supporting networked games, and the methods used to estimate subjective quality of the game users. There is a special difficulty with this kind of service, since gamers show a very demanding profile: they always want the best in terms of graphics, speed, frames rate, sound, etc. As far as network is concerned, gamers have also proven to be very difficult customers to deal with: in (Chambers et al., 2005) different game traces from a number of servers were studied and some conclusions were devised: players are impatient when connecting to a server, they present short attention spans, they are not loyal to a server (when they can select between different servers of the same game), and they reveal when they lose interest, so game providers can detect waning interest and react to it. Some consequences of this are that game workloads are only predictable over short-term intervals. Finally, predicting the success of an online game has revealed as a very difficult task, so the provision of the resources is a complicated problem, which is stressed at launch time of a new title.

The impact of network on the shared virtual world

As previously said, when the whole game is located on a local device, be it a PC, a console or something else, a player can interact with the virtual world of a game in "real time" meaning that the time which passes since the player makes an action until the virtual world presents a response to that action is very low –lower than the limits of human perception, usually just a couple of milliseconds). However, when we introduce the notion of a shared multiplayer virtual world, we also introduce additional latency in receiving the response, caused by the network. We will illustrate this using a simple case of a duck-hunting game using a client-server architecture with three clients (Fig. 1). We will consider that each of the three clients experiences 50, 100 and 150 milliseconds of one-way network delay from their computers to the server respectively. Let us assume that the second client shots the target at a certain moment. The time it takes the action to travel to the server and back is 200 milliseconds. Thus, at 100 milliseconds mark, only the server will know that the duck has been shot; at 150 milliseconds, client 1 will know that the duck has been shot, and at the server side the dog has already retrieved the duck. At 200 milliseconds, client 2 will see he has hit the duck, while client 3 still sees the duck alive.

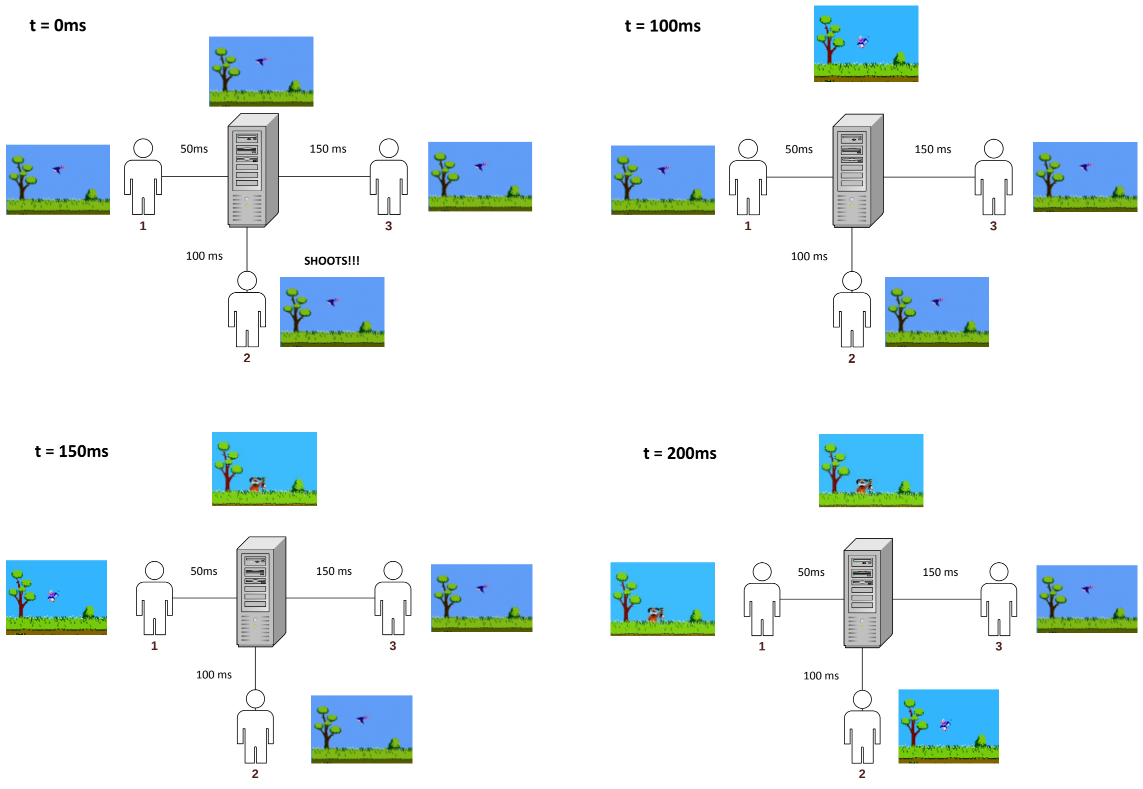


Fig. 1 Scheme of a game with three players experiencing different amounts of latency

There are two basic approaches for handling the synchronization of the state of the distributed virtual world. The first on is "lock step" deterministic simulation which ensures that all clients have the same state through delaying the execution of commands for a time period which is needed for all the participants in the virtual world to get the information regarding that command. Such approach is used, for example, in *Starcraft 2*, where all the actions of the users are delayed for the default value of 200 milliseconds. This approach is viable in games which lack very high and strict latency requirements, such as strategy games. In our example this would increase the response time from action to reaction of client 1 for 200 milliseconds, which is higher than the latency imposed by the network itself. The second approach is to execute the updates as soon as they arrive to the client, which is employed in very fast-paced games like First Person Shooters. In our example this approach would increase the delay in which reaction is received by the client 2 by the value of network latency, but this approach also creates three different versions of the state of the virtual world, which are separated in time on the different client machines. These discrepancies between client side states of the virtual world can cause unexpected behaviors, anomalies, and reduction of the Quality of Experience of the end users. As we will see, game developers employ various mechanisms to deal with these anomalies.

As it can be deduced from the example, the worst enemy of gamers, from a network point of view, has a name: *latency*; and this is stressed for the most interactive game genres, where movements are fast, and some tens of milliseconds do matter. This problem is shared with other real-time services and different means for reducing it are being proposed. In fact, the Internet Engineering Task Force (IETF), in charge of developing the Internet protocols, is looking for proposals and organizing meetings with the aim of reducing the latency of the Internet (Ford, 2014).

Online Games Classification

Online Games Genres

Games can be grouped into different genres, based on their mechanics and concepts. While game genres are quite general terms with a bit blurred border, certain tendencies regarding the network traffic of each genre can be observed. The NPD Group, in its *"Software Category Definitions"* report (2008), established 13 different game categories. However, not all these categories include the possibility of playing online. In the present chapter, we will only focus on those categories where online playing is a must, or those where networked versions of the games are usual. According to this, the game genres considered are:

- First Person Shooters (FPS): Is a game genre which yielded the first multiplayer online game *(Maze war)* and was one of the first genres which became popular worldwide with *Wolfenstein 3D*. In a FPS a virtual world (often very limited in size) is shared by some tens of players grouped into teams, each of them controlling an avatar, with the aim of accomplishing a mission or killing all the enemies. The view of the user is first-person, meaning that he sees the hands and the weapon of his avatar, and the part of the scenario in front of him. The main characteristic of this genre is the high interactivity it requires: movements and shots are really fast, and the aim of the player matters. For example, average Time To Kill (TTK) or average time since the weapon is fired until the opponent is slain, in a recent FPS (*Call of Duty: Ghosts*, released in 2013) is just 161 milliseconds for the most popular assault rifles. Studies have shown that these games need very low latency values for providing a satisfactory QoE.

- Massively Multiplayer Online Role Playing Games (MMORPG): In these games the virtual world is usually very big, and the number of real users sharing it is also huge (typically several thousands). In addition, a number of NPCs (Non-Player Characters) controlled by AI (Artificial Intelligence) are present in the virtual world. Each user controls a persistent avatar which can learn new abilities and earn weapons and equipment. A wide range of activities can be performed in the game: different quests assigned by NPCs allow the player to improve the skills of his avatar; trading with other characters is also allowed, using virtual money; group missions are included, where players have to cooperate in order to accomplish a difficult challenge; finally, some areas in a virtual world permit a number of players to fight against another group. Although the interactivity of these games is not as critical as in FPSs (e.g. the aim is not as important, since you first click on the objective and then select the spell or weapon to use against it), the speed and the delay also matter and can decide the result of a fight.

- Real Time Strategy (RTS): A number of players (typically up to 10) share a virtual world, and each of them controls a civilization including army units, buildings and structures. The typical actions of the player include gathering different resources, making new buildings, technological development or creating armies. The control of the virtual characters is indirect, meaning that the player only selects the characters and assigns tasks to them (e.g. gathering food, attacking an enemy town, etc.). The name "real-time" means that they are not turn-based, but all the players act at the same time, each one from a computer.

- Multiplayer Online Battle Arena (MOBA): In these games, considered as a subgenre of RTSs, two teams fight in order to conquer the battlefield. Each player controls a character and has to cooperate with other real players, and also with other ones controlled by the computer. The control is more "direct", taking into account that only a virtual avatar is directly managed by the player.

- Sports: This genre groups very different games, including car racing, soccer, etc.

Delay sensitivity of different genres

As remarked in (Feng et al., 2005), gamers are very sensitive to delay, but the maximum tolerable delay varies with the game genre. Thus, some studies have taken a simple approach, just based on latency, and ignoring other network impairments as jitter (i.e. variation of the network latency) or packet loss. This can be enough in a first approach, although some more advanced models for estimating subjective quality also consider other network impairments, as we will see.

In (Dick et al., 2005) a survey with a number of FPS games was carried out. As a result, it was reported that a one-way delay of 80 milliseconds could be acceptable for most of the users (they rated it as "unimpaired"). In (Henderson and Bhatti, 2003) it was shown that delay has an impact on the decision of a player to join a game server, but the influence on the decision of leaving the server is lower.

Regarding MMORPGs, the study carried out in (Ries et al., 2008) showed that the quality level rated by the players, dropped from "excellent" to "good" for one-way delays greater than 120 milliseconds. In (Chen et al., 2006a) it was reported that the duration of game sessions declined for values of the one-way delay about 150 or 200 milliseconds. Some studies have also been carried out for RTSs: the reported delays range from 200 milliseconds (Cajada, 2012) to 500 milliseconds (Claypool and Claypool, 2006) of one-way delay.

A final remark has to be added here: the game genre is not the only parameter to be taken into account when estimating the maximum delay that a player may tolerate. Some studies have shown that the experience of the player increases his/her delay sensitivity, i.e. an experienced player gets annoyed more easily than a novel one (Suznjevic et al., 2013). In addition, the behavior and the skill level of the other players in the party also have an influence on the subjective quality. All in all, the figures presented have some degree of uncertainty, and may not only depend on network latency.

Architectures of online gaming systems

The synchronization of a game is not a trivial problem: all the clients share a virtual world and it must be consistent, taking into account that the network latency experienced by each player may have a different value, and it may also vary during the party.

Two basic models have been in use for online games: client-server and peer to peer (p2p). In the client-server model there is one central entity which holds a "true" copy of the virtual world state (usually a server) and all the other entities are under its authority. As a consequence of network latency, desynchronization may occur, and it may happen that the game state calculated by a client is corrected by the server. In peer to peer games this central authority does not exist, so clients have to synchronize between them. In the history of online games both models have been used, but in the last decade a clear dominance of the client-server architecture can be observed (Feng et al., 2005). There are some exceptions where a peer-to-peer scheme is used, as e.g. *StarCraft 1* (1998), where up to 8 people could play together (Claypool et al., 2003; Lee, 2012). Some academic research also considered the peer-to-peer model for an experimental game called *MiMaze* (Gautier and Diot, 1998). In addition, the possibility of using a peer-to-peer architecture for a popular MMORPG *(World of Warcraft)* was considered in (Miller and Crowcroft, 2010), but the results showed that it was not a good idea, since it would occasionally saturate residential connections and would increase latency with respect to client-server solutions. In the remainder of the text we will focus on the client-server model.

Client-Server model

There are several reasons for companies to employ client-server architectures. First of all, the fact of having a single "authority" makes synchronization easier. The server is the authority and the clients must obey. However, cheating in online games is also possible: for example, if the timestamp of an action is modified before a packet is sent to the network, a shot can be considered as previous with respect to the enemy's one, thus giving an advantage to the cheater. In that sense, a server establishing a single time source can avoid the problem. Another clear advantage appears, i.e. billing is easier if the company controls the server: if the player does not pay (or does not register properly), he/she may not be allowed to play. Companies employ different policies regarding the control of the server. They can be classified as:

- The server application is distributed with the game (either included in the game itself or as a separate application) and can be installed and controlled by the player (e.g. *Warcraft 3, Counter Strike 1*). The user is then able to create a dedicated server, or even one of the clients may act as the server. This option allows the users to create LAN parties, where latency can be really low, or to host the games over the Internet, i.e. users may create a high number of servers (with public IP addresses) on which other users can play. If this policy is employed, the game developer does not have full control, but neither has to invest in the hardware and network infrastructure. Over the time different services for finding available servers have been developed (e.g. gametracker.com) and they significantly help the players to find servers which satisfy their demands for network latency, game type, etc. This option was used by the authors of (Feng et al., 2005) to obtain detailed statistics of *Counter Strike 1*, generated in a public server created by them.

- The dedicated server is included in the released application, but it is not controlled by the user. This means that any player may act as the server, but the game developer company performs a sort of "orchestration" between the users, selects the one with the best connection, and assigns his/her client application the role of the server. This process is transparent to the player, who may or may not act as the server, and is not aware of that fact. This has a clear advantage for the company: they only have to join a number of users who will play together, but the processing and network charge required for supporting a party is "externalized" and assigned to one of the players. As a result, the scalability is clearly increased, but this has a counterpart: what happens if the user acting as the server leaves the party? Fig. 2 shows a traffic capture of *Call of Duty* for *Playstation 2* obtained in May 2012, where an interval when the game stopped can be appreciated. At *t*=500 seconds, the player who is acting as the server *(server #1)* decides to leave the party, so there is a gap until a new player is selected to act as the server. During the gap, the rest of the players watch a *"server migration"* message in their screens, and they have to wait until the device of another player takes the role of the server *(server #2)*. Another problem of this policy is related to the unfairness: the player acting as the server has a clear advantage, because he experiences null network latency.

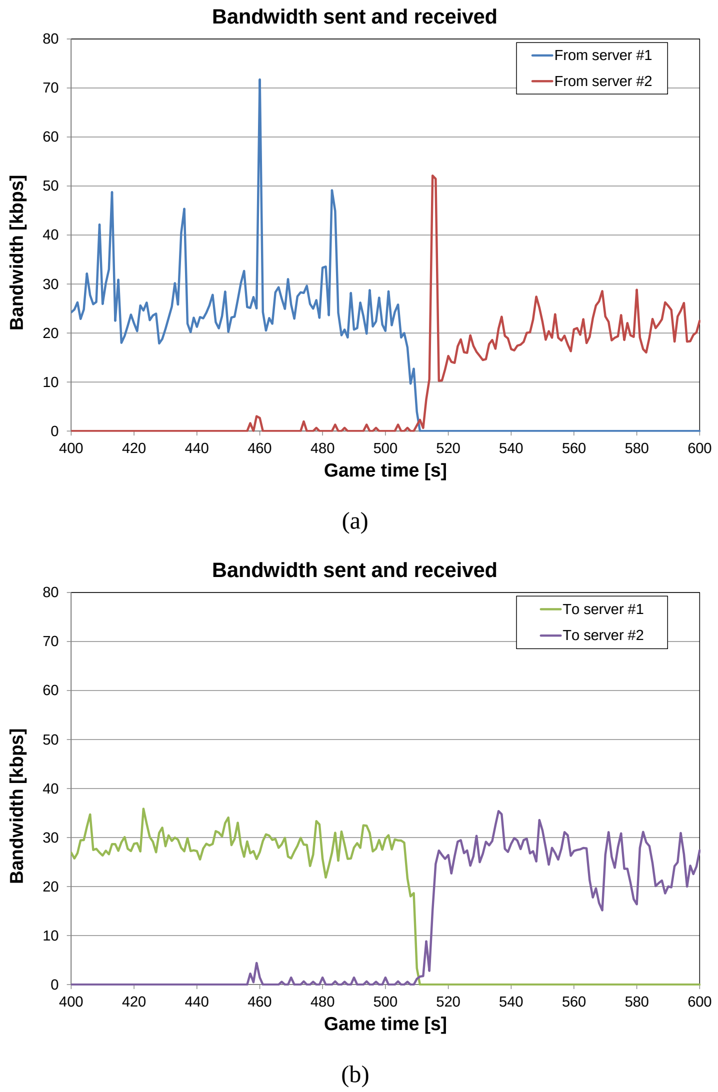


Fig. 2 Traffic capture during a *server migration* in *Call of Duty*

- Server fully controlled by the developer/publisher (e.g. *World of Warcraft*). This is the most common tendency nowadays for certain types of games such as MMORPGs. The servers are placed, controlled and maintained by the game company which enables complete control, cheating prevention and advanced business models (e.g. selling virtual items for real currency). There have been some cases in which reverse engineering techniques have been employed in order to decrypt the traffic, or to create a private version of the whole game server (this has happened for example for *World of Warcraft)*. This is also known as "server emulation" and has many legal issues (Debeauvais and Nardi, 2010). However, these versions usually do not work with the last versions and updates of the client and may contain bugs and viruses.

- In some Real Time Strategy games (e.g. *Starcraft 2),* a peer to peer model is employed, but there is still a server. The server acts as a traffic aggregator: it receives the traffic from each client and sends the aggregated input to the other ones. It does not hold the game state (so there is not a central authority), but it forwards the game commands received from each player to the others. *Starcraft 2* is based on lock step deterministic simulation: the method is based on each client queuing the commands received from every other player, and executing them in the future (12 frames, which may be equivalent to 200 milliseconds). The advantage of this communication approach between the clients for RTS is clear: they require very low bandwidth, since they just send the player's commands, instead of sending the updated position of each unit (there may be thousands of them). The presence of the server also reduces the sent bandwidth, since each client only sends its player's commands once, and the server is the entity in charge of forwarding them to the other clients. As a counterpart, an observable input delay appears, since units do not respond immediately to player's commands. In addition, the slowest player (i.e. the one with the highest latency) slows down the game for all of them. The reasons for including a server in *Starcraft 2* are not only related to traffic scaling. The server is also in charge of authentication, storing player's data, matching players with similar skill levels, and performing anti-cheating mechanisms. It is also able to reduce piracy, since the players have to use the server.

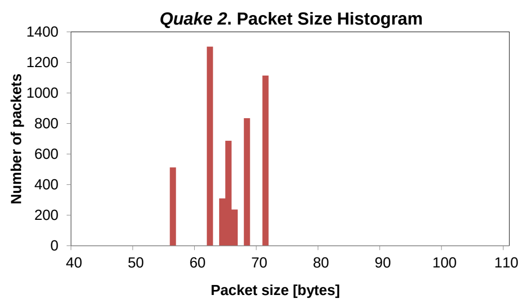
General Characteristics of Game Network Traffic

The main functions needed for executing an online game can be divided into: gathering the player's commands, executing the logic of the virtual world based on those commands, rendering the virtual scene and displaying the information to the user. The information which is transferred between the client and the server is dependent on the location of these functions. In the majority of traditional games, the game logic is executed on the server while the remaining functions are done on the client. In that case the information traveling from the server to the client comprises updates of the virtual world state which are then rendered and displayed on the client. From the client, only the commands of the player are sent to the server. In a more recent approach, commonly called "cloud gaming", both game logic and virtual world rendering are located on the server, from which a high quality video stream is sent towards the client, while the player's commands are transferred from the client to the server.

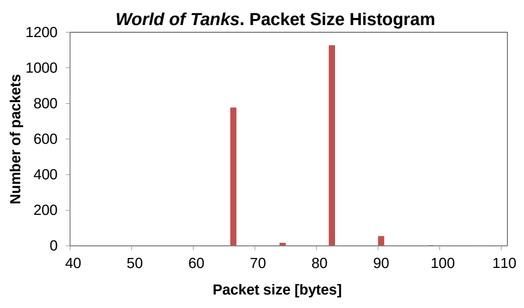
In addition to player commands and inputs, and virtual world state refreshes and video streams, other flows may appear, as chat or audio for player communication, authentication, accounting, etc. Although some games have in-built VoIP systems, many players use standalone applications (e.g. *Teamspeak*, *Ventrilo*, *Skype*) when playing. Finally, game updates, and downloads of elements as maps or some textures created by the players may also require additional traffic flows. In traditional online games the game content (scenarios, characters, textures, etc.) has to be stored in the computer of the player in order to avoid its sending during the game. This often results in high storage requirements in the hard disk of the user. Two examples: The *World of Tanks* v9.2 folder requires 28 Gigabytes, whereas *World of Warcraft* v5.4.8 uses 24.8 Gigabytes. Game clients and supplementary data have been traditionally distributed over CDs or DVDs, but today it is also common that the game clients are just downloaded from the Internet. These flows in which game clients are obtained are not real-time, but just typical Internet file downloads. However, these flows are not the ones directly supporting the game action, so they are out of the scope of this chapter.

Network Traffic Characteristics of Traditional Games

As said in the introduction, game providers control the two extremes of the communication: they manage the server and they develop the game client application. However, they do not have a strict control over the network connecting both elements (usually the Internet). So they have to drastically limit the traffic to be exchanged during the game to what is strictly necessary, looking for a wide penetration, even in countries where bandwidth is scarce. This tendency has not changed over the years: Fig. 3 and 4 show the traffic profile of two First Person Shooter games (which are considered, traffic wise, the most demanding game genre): 5000 packets of a client-server flow of *Quake 2* (1997) are compared with the traffic of *World of Tanks*, using 2000 packets captured in 2014. As it can be seen, high rates (inter-packet time of about 30 – 50 ms) of small packets (60 – 80 bytes) are employed in both cases. The bandwidth of a client-server flow is roughly 16 kbps for the former, and 8 kbps for the latter. A high number of game updates are needed in order to maintain the illusion of the virtual world which is responsive in the real time (i.e. below the time for which the human perception will notice the delay). In general, for the high Quality of Experience the networked aspect of the game should be hidden from the player and the game should run like it is stored on the local computer.

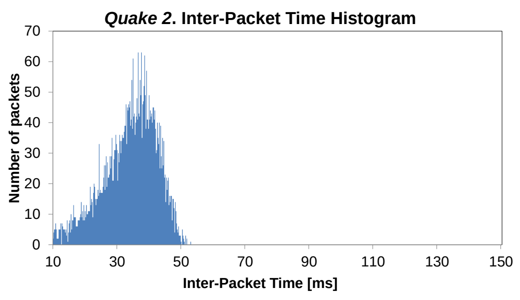


(a)

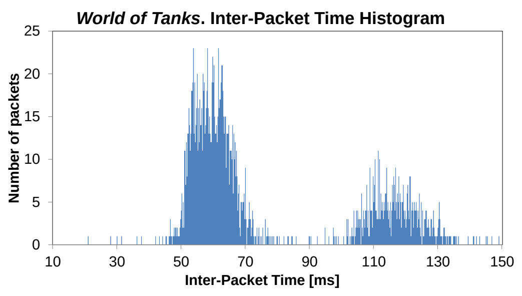


(b)

Fig. 3. Packet size histogram of a) *Quake 2* (1997) and b) *World of Tanks* (2014)



(a)



(b)

Fig. 4. Inter-packet time histogram of a) *Quake 2* (1997) and b) *World of Tanks* (2014)

Thus, as illustrated in Fig. 5, the typical structure of an online game consists of a number of client applications and a server. During the game, no information is directly exchanged between the clients. Each client sends a flow to the server including the movements of the gamer. The server calculates the next state of the game and sends it to each player. In this scenario three “bottlenecks” can be identified, namely:

- The uplink resources, which must be enough so as to permit the information of each gamer to arrive in time to the server.
- The processing capacity of the server, in charge of calculating the next state of the game.
- The downlink capacity, necessary to send the state of the virtual world to each player.

Two of these bottlenecks are related to the network, which begins with the access connection of the player, includes a number of Internet links and ends with the connection of the server. The most stringent link is most likely the access link of the player, assuming that the game provider has correctly dimensioned his network.

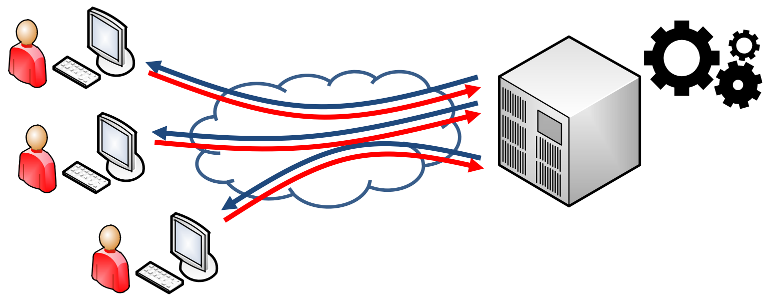


Fig. 5 Client/server infrastructure and traffic flows exchanged during a party

There is a scalability issue derived from the use of a client/server architecture: while the amount of information received by the server grows linearly (if the server manages a party with *N* players, it will receive a flow reporting the movements of each one), the amount of information to be generated by the server grows with the square of *N*. Certainly, the server has to report to each of the *N* players the actions of the rest (*N*-1) of the players, so the number of flows is *N*•(*N*-1). This was the reason behind the limitation of the number of players in First Person Shooters to some tens: the downlink of residential connections. As reported in (Feng et al., 2005), some of these games were *"designed to saturate the narrowest last-mile link"*. As we will see, different techniques can be employed to overcome this limitation in the genres where the number of players sharing a virtual world must be higher.

Traffic profile: High rates of small packets

Despite their different genres, networked games share a set of common characteristics. The first one to be highlighted is that online games generate long lived flows. Nevertheless, the duration of the session depends on the game genre. In (Feng et al., 2005), the traffic of a First Person Shooter server was analyzed, and the results showed that the typical connection of a player lasted about 30 minutes in a server. At the same time, 99 % of the sessions lasted less than two hours. In contrast, an MMORPG was studied in (Tarng et al., 2008), and the average session time measured was 2.8 hours.

According to their real-time requirements, they tend to generate traffic profiles consisting of high rates of small packets, and this is stressed with the interactivity level of each genre. According to this, First Person Shooters are usually the ones presenting the highest rates and the smallest payloads.

As said in the introduction, the traffic profile based on high rates of small packets does not result in a high bandwidth usage. Many of these games can be played with a residential connection of some tens of kilobits per second. The "historical reason" (i.e. a 33 kilobits per second modem should suffice to play the game in the 1990s) still persists since, regrettably, this may be the bandwidth amount available in some developing countries nowadays. This low-bandwidth usage is shared by FPSs (Ratti et al., 2010; Feng et al., 2005), MMORPGs (Svoboda et al., 2007; Suznjevic and Matijasevic, 2012), MOBAs and RTSs, which bandwidth usage is even lower: from 4 to 6 kilobits per second, according to (Claypool, 2005). In the case of MMORPGs, the bandwidth may strongly vary depending on the activity performed by the player. Some studies have divided player actions into different categories (e.g. *trading, questing, player vs. player)*, and the difference in terms of bandwidth can be up to 5 times (Suznjevic et al., 2009). If the player is fighting, his/her application is expected to generate more bandwidth than when he/she is just trading (buying or selling certain virtual goods). This is also related with the number of players involved in the action: the server has to send more information to the client if the number of players surrounding his avatar is high.

These usually low bandwidth requirements increase the market penetration of networked games, which are not only targeted for affluent countries. In fact, in developing countries many people who do not have Internet connection at home still play these games in Internet cafés, very popular in these areas (Furuholt et al., 2008). In (Batool and Mahmood, 2010) and (Gurol and Sevindik, 2007) the profile of the users of Internet cafés was studied, reporting that more than 50 % of them do play online games there.

Another characteristic of these traffic flows has to be highlighted: they are very inefficient in terms of ratio of useful (game) information and signaling information (headers). Taking into account that an IPv4/UDP (the Internet Protocol version 4, and the User Datagram Protocol) header requires 28 bytes, if many of the payloads are about 20-60 bytes long, it can be seen that the efficiency is really low. This is the reason why some optimization techniques, based on header compression and multiplexing, have been proposed (Saldana et al., 2013), as an adaptation from VoIP optimization standards (Thompson et al., 2005).

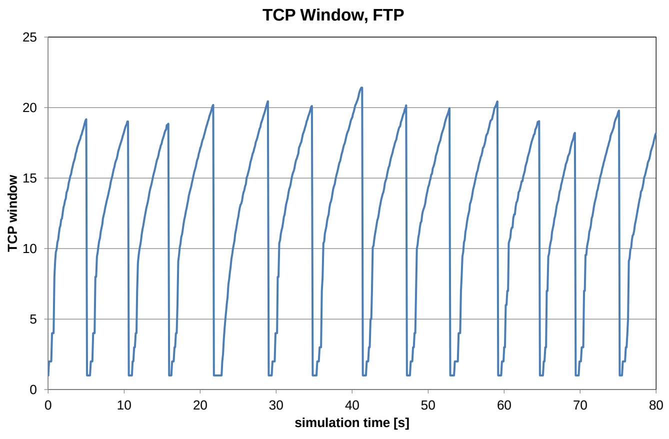
Network protocol: TCP or UDP?

UDP is more suitable than TCP (Transmission Control Protocol) for real-time services. UDP just sends the data and does not expect any feedback from the destination. In fact, RTP (Real-Time Protocol), the protocol used for sending VoIP streams, is designed for traveling on UDP datagrams. So in a first approach, the most intuitive option for an online game would be to select UDP at the transport level. In contrast, TCP is a closed-loop protocol initially designed for bulk transfers: it tries to get the maximum bandwidth amount, while maintaining fairness and avoiding the saturation of the links. For this aim, it integrates a number of mechanisms, which rely on the reception of Acknowledgement (ACK) packets from the destination, in order to control the transmission rate and to retransmit lost packets.

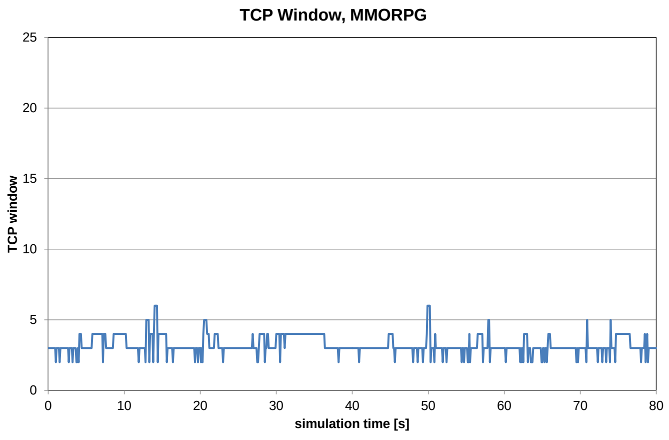
UDP prioritizes the continuous arrival of packets against the reliability of TCP. The disadvantage of UDP is that the service has to be designed with some tolerance to packet loss. As it happens in VoIP, where the possibility of losing some samples is assumed, and robust codecs are designed with the capacity to interpolate and regenerate lost information, UDP-based games have to implement some policies for hiding packet loss to the player. We will discuss them later.

However, some online games do use TCP, and this is somewhat related with the game genre. Some clear trends can be observed: FPSs always generate UDP flows. Many MMORPGs use TCP, although some of them use UDP (Chen et al., 2006b). The most popular MOBA (*League of Legends*) uses UDP. RTSs may use TCP or UDP.

When a game uses TCP, the initial assumption behind this protocol, *"I have to transmit this information as fast as possible"* is no longer true, taking into account that the information to be transmitted does not exist previously, but is generated as the user plays. This makes the behavior of TCP very different from what could be expected. In the literature, the normal use of TCP (e.g. for downloading a file) has been described as *network-limited*, since the network is the element setting the limit, whereas the use of TCP for an online game is called *application-limited* (Wu et al., 2009), taking into account that the application generates a continuous flow requiring a certain amount of bandwidth. The difference is illustrated in Fig. 6, where the evolution of TCP sending window (which controls the transmission rate) is shown. In *a)*, it can be observed that, during a file download (simulated in NS2) using FTP (File Transfer Protocol), the window follows a *sawtooth* shape: TCP increases its rate until network congestion occurs, and then it folds back, and begins the increase again. In contrast, *b)* has been generated with the traffic model for *World of Warcraft* (an MMORPG) presented in (Suznjevic et al., 2014), and it can be observed that the size of the sending window is always low. There are even certain moments where TCP has nothing to send.



(a)



(b)

Fig. 6. Sending window of TCP for *a)* an FTP download; *b)* a client-server flow of *World of Warcraft*

Additionally, the use of TCP stresses the inefficiency of game flows. In (Svoboda et al., 2007) it was reported that these games set to 1 the PUSH bit of the TCP header, thus making the protocol send the packet as soon as possible, without waiting for having a full payload. A bidirectional TCP connection is established between the client and the server, and ACKs are piggybacked into normal packets. However, in the traffic traces used in the study, 56 % of the packets were *pure* ACKs, i.e. they had no payload.

Client-based vs Cloud Games

The increase of the bandwidth available in households in some countries has led to a new model for supporting computer games, namely *cloud gaming*. In this model, the game is run on a remote server (both game logic and virtual scene rendering), and the client is no longer a big application, but just a thin one in charge of reproducing a video stream and transmitting the player commands to the server. This has some clear advantages:

- Any game can be played without a previous installation. The model can be "select and play".
- The hardware requirements of the client device are drastically reduced, since it only reproduces a stream generated by the server (Claypool et al., 2012).
- From a business point of view, the subscription model is clearly the most suitable one. It should be noted that the player can not only play multiplayer games, but also individual ones, so he/she will even pay for using an individual game online.

As a counterpart, the server infrastructure to be deployed by the game distributor is bigger, since each frame to be sent to a player has to be calculated on the server side. Also, bandwidth required is obviously higher, typically higher than 3 Mbps which is much higher than traditional games. Fig. 7 shows a comparison of bandwidth usage of various genres of traditional games and a game implemented as a cloud game. Cloud gaming can require up to 1000 times more network bandwidth than traditional games!

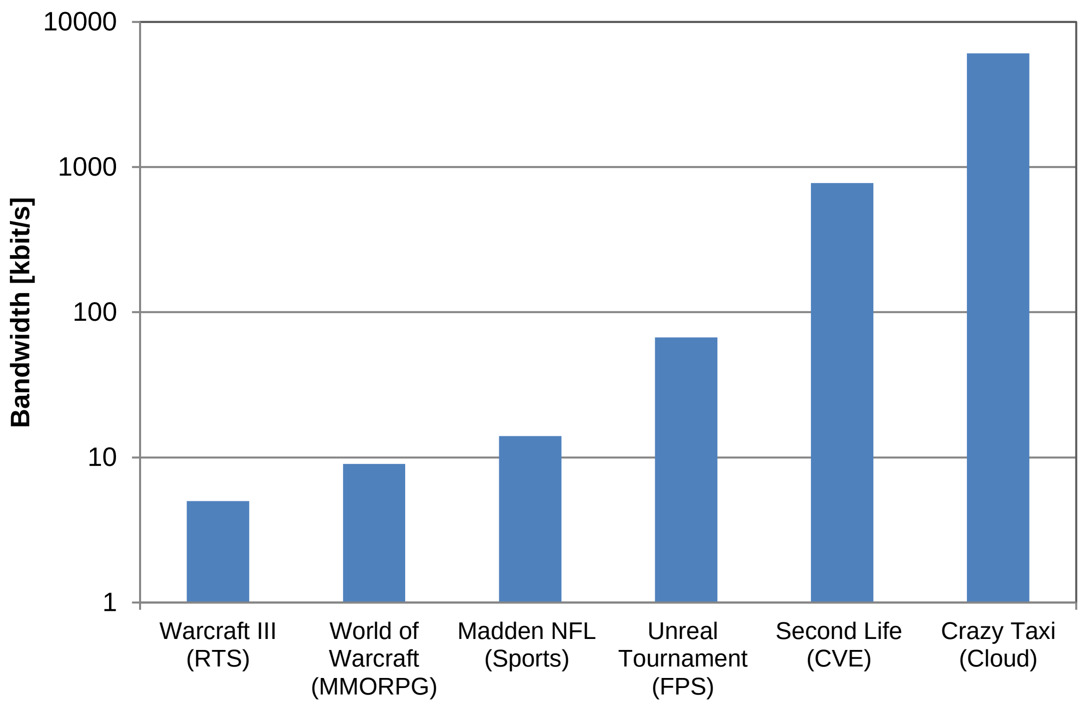


Fig. 7. Bandwidth usage of traditional game genres and cloud game

Estimating Quality of Experience

Quality of Experience (QoE) has been defined by the COST action IC 1003 Qualinet (Qualinet, 2013) as the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personality and current state. QoE has many aspects that comprise it and many factors which influence it. Influence factors can be grouped into *system*, *context* and *user* factors. Games are a specifically hard topic for QoE research due to the complexity of the playing behavior and the additional factors which have an impact on the QoE (e.g. players' expertise in the game). This work is focused on one subset of system factors - network parameters, but it should be taken into account that many other factors impact the final QoE.

It can be said that VoIP was the first real-time service which became popular on the Internet. In order to accomplish its real-time requirements, it has to send small packets with a high cadence. The round trip time (the time a packet takes to go to the other side and return) has to be kept low in order to make it possible for two people to maintain a normal conversation without interrupting each other. The one-way delay is also known as *mouth-to-ear* latency, and it not only includes network delay, but also the latency caused by the codec (e.g. 20 ms for including two samples) and the equipment. The ITU defined a standard way to measure one-way transmission time in their G.114 Recommendation.

Online games' traffic has many similarities with that of VoIP: they also generate small packets with high cadences, so a comparison can be established between mouth-to-ear latency and player-to-server delay. Players usually talk about "lag" or "ping" as the round trip time. This parameter is of primary importance for them and in fact, many games (e.g. *Counter Strike)* include the "ping" in the score table, in addition to the hits achieved by the player.

When defining the quality of a voice call, the ITU not only tried to capture objective parameters as e.g. delay or loss of information, but it also developed a model able to estimate the subjective quality, taking as an input the objective parameters. The E-Model, defined by ITU-T Recommendation G.107, calculates a score, called the "R factor", ranging from 0 to 100, which can be directly translated into a Mean Opinion Score (MOS) from 1 (bad) to 5 (excellent) (Cole and Rosenbluth, 2001).

The difference between games and voice is clear: a conversation is always similar, because the problem is to transmit the sounds from the speaker to the hearer in a clear and non-delayed way. However, when considering a game, many factors intervene:

- The interactivity requirements of the game: As we have seen before, each game genre has its own characteristics and delay requirements. For example, in a First Person Shooter, a delay of 300 milliseconds may jeopardize the game, whereas this delay can even be not noticeable in a Real Time Strategy game.
- The way the game is developed: The implementation of different mechanisms (we will discuss them later) may hide the effect of network impairments to the player. Some advanced techniques, as smoothing the transition between the state predicted by the client and the real state received from the server, may significantly reduce the player's nuisance caused by delay.
- The transport protocol employed: If TCP is used, no information will be lost between the client and the server. However, if packet loss occurs in the network, this will require retransmissions and may cause additional latency: it may happen that one TCP packet is lost in the network but subsequent ones arrive correctly. In this case, the packets already arrived will be stopped by TCP till the new copy of the lost packet arrives, since TCP has to deliver packets in order.

The development of a subjective quality model requires a number of real people to participate in tests with the service. There are two basic methodology approaches which can be used: *a)* single stimulus, in which for one gaming session players give an Absolute Category Rating (ACR) of QoE for some specific parameter values (e.g. MOS tests), or *b)* double stimuli in which the users compare playing sessions with different parameter settings and decide which one is better (e.g. paired comparisons). Also, there are approaches in which bodily reactions are measured to establish QoE values (e.g. EEG, face expression etc.). Therefore, a single model able to produce a subjective quality score for all online games, using as input the network parameters, cannot be developed. However, some specific models have been developed for certain games (Ries et al., 2008; Wattimena et al., 2006), and some more generic models have been proposed (Ubicom, 2005).

Other approach some studies have taken is to perform "objective measurements" (Kaiser et al., 2009): a number of identical "bots", i.e. virtual avatars controlled by Artificial Intelligence, are placed in the same virtual scenario and a number of parties between them are performed. If the number of parties if high enough, then the score will be the same for all the bots. Then, different network impairments (latency, jitter, packet loss) are added to one of the bots, and another set of tests is performed. The performance degradation of the network-impaired bot can then be statistically characterized. However, in this chapter the focus is on estimation of subjective quality of real players.

Network Impairments Considered

When network operators post their offers, they usually summarize everything into a single parameter: the maximum connection throughput measured in megabits per second. This is a parameter everyone can understand, and it is of primary importance when using the Internet. However, as long as online games are concerned, the maximum throughput may not result the most important parameter: as said before, online games do not require a high throughput (we are not talking about cloud-based games).

These network impairments are often known as "Key Performance Indicators (KPI)". In online games, the three mainly considered are latency, jitter and packet loss. Between them, it is clear that the most important one is latency due to jitter and packet loss have small values in today's networks.

These impairments are strongly related between them, and they have sometimes been studied together for many online games. For example, the effect of delay and jitter was studied in (Dick et al., 2005) for two FPS, a sports and an RTS game. Its effect is very different depending on the considered game and genre.

Latency

Latency can be defined as the time required for transmitting player's information from the application's layer of the client to the application's layer of the server. This would be the one-way delay. If the inverse path (from server to client) is also considered, then we have the round-trip time, informally known as "ping" by computer gamers. Many games (if not all) measure the value of the ping and report it to the user, taking into account its crucial importance when playing an online game. It is usually the only network parameter presented to the user.

If the latency is high, the virtual world is not responding in real time and the player perceives his actions being "late". This severely impacts the QoE of the player and his immersion in the virtual world. The situation gets even more complex when we have multiple players with different latencies. These differences in latency can cause inconsistencies and anomalies in the virtual world, the most known being "shooting behind the wall problem". Let us assume that a player with a high delay is just running to hide from an enemy with lower latency, and the enemy shoots him while he is running across the open space. While the information about that shot reaches the player with high latency, in his section of the virtual world, he will have some time to e.g. go round a corner, before the "you have been killed" message from the server arrives. So the perception of this lagged player will be something like "I have been shot around the corner which breaks the laws of physics". We will illustrate this problem in the section about QoE-enhancing mechanisms included in games.

When a networked game is played in a LAN, latency can be kept really low, in the order of 1 millisecond. In fact, some players organize the so-called "LAN parties" in order to play together with high bandwidth and also low latency levels. Some of these LAN parties are small, with a group of people going to other's houses and bringing their own computers. In Asia the games are, in general, played in large Internet cafes which also have their own LANs and the games are often played in those LANs. Other LAN parties can be organized as big events with hundreds (or even thousands) of players in the same place for some days (e.g. a weekend), where public competitions and tournaments may also be held. The possibility of playing with really low latency levels is also a factor attracting people to these events. One of the biggest LAN parties is *Dreamhack*, held twice a year in Sweeden. *Dreamhack* holds the world record for the largest LAN party in the world with over 22,000 players, and has also held a record for the fastest Internet link in the world (120 Gigabits per second) up to 2012 when it was beaten by the second largest computer festival in the world: The *Gathering* in Norway.

However, the problem we are considering in this chapter refers to the case where networked games are played through the Internet, with each player running his/her own computer at home. This case covers the vast majority of the gaming activity in the world. In this case, the concept of "latency budget" can be used (Ford, 2014), taking this approach: there is a limit in the maximum delay a player is prone to tolerate, and there are different "sources of latency" contributing to it. These sources can be divided into the next categories:

- Generation delay: It refers to the delay between the physical event and the availability of the data. In the case of an online game it is negligible, since the event to be captured is a key stroke, the movement of the mouse or the game controller, and it can be assumed that they are almost instantaneous.
- Packet transfer delay: It accounts for the propagation of the information between the source and the destination. It first includes the propagation delay, which refers to the part caused by the speed of light: for example, if the distance between the client and the server is 5,000 km, an unavoidable one-way delay of 16 milliseconds appears. This gets increased in optical fiber: since its index of refraction is higher, the speed of light is reduced to about 200,000 kilometers per second. Other contributions to transfer delay are buffering in intermediate routers, and the time required for sending the bits at a certain rate (transmission delay). This may be negligible on Gigabit links, but not in a residential access (e.g. a DSL uplink of 1 Megabit per second would require 12 milliseconds for transmitting a packet of 1,500 bytes). All in all, packet transfer delay not only depends on the distance, but on the number of intermediate routers, the congestion level of the traversed networks, the bandwidth of the links and even the size of the packets.

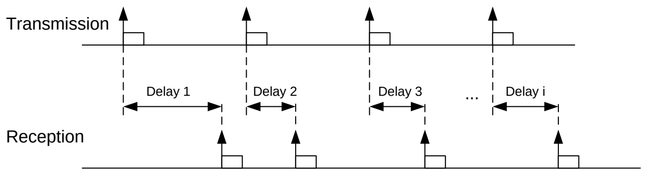
- Processing delay: In online games, it is mainly caused by the response time of the server. This can be small in client games, but it may be significantly higher in the case of cloud games, since the server has to render the scene, code it in a video and send it to the client application, which has to decode and display it to the user (this subpart can also be called “playout” delay). In (Huang et al., 2013) a comparative study including two commercial cloud game platforms was presented, and the processing delay was in the order of 100 to 300 milliseconds.

Many actions are being deployed in order to reduce or mitigate the contributions to the latency, some of them more effective than others. For example, the problem of “bufferbloat” (Gettys and Nichols, 2011), i.e. the excessive delay caused by over dimensioned router buffers is being issued in the last years through mechanisms as Active Queue Management (AQM) (Adams, 2013). Geo-location of content sources (game servers in our case) is also a method for drastically reducing latency.

Variation of latency

The statistical variation of the delay between different packets of a flow is sometimes known as “jitter”. It may be caused by network congestion, which is translated into a different queuing delay for each packet on a flow. Another source of this variation can be that different packets of the same flow may not follow the same network path. If the traffic of an online game shares the bandwidth of a residential connection with e.g. a TCP file download, this may also cause high delay variations, taking into account that TCP throughput typically follows a sawtooth pattern (see Fig. 6a).

Different definitions of the variation of latency can be found in the literature. In RFC 3393 the IETF defines the Inter Packet Delay Variation (IPDV) for two packets inside a stream as *“the difference between the one-way-delay of the selected packets”*. In RFC 3350, the method for measuring the jitter between each pair of RTP packets is defined: *“The interarrival jitter J is defined to be the mean deviation (smoothed absolute value) of the difference D in packet spacing at the receiver compared to the sender for a pair of packets”*. In *netem*, a network emulator included in Linux, the jitter is expressed as the standard deviation of the delay, following a normal distribution. In (Brun et al., 2006) the end-to-end jitter is formally defined as the *“expected absolute value of the sum of inter-packet delay variations introduced by each node along the path between the source and the destination”*. The definition of Inter-Packet Delay Variation (IPDV) is illustrated in Fig. 8.



$$IPDV = \frac{\sum \left| delay_{i} - delay_{i-1} \right|}{n-1}$$

Fig. 8. Definition of Inter-Packet Delay Variation (IPDV)

In Voice over IP, the jitter is compensated with a playout buffer where packets are buffered and played in a steady manner. Thus, the jitter is translated into an additional delay and packet loss, which happens when a packet arrives too late to be played. In fact, the formula for calculating the R-Factor of ITU’s E-model does not consider jitter as an input. Something similar happens with video streaming or Internet TV, but the buffer can be in the order of seconds in these cases, since they do not present the same interactivity requirements.

Taking into account the tight interactive requirements of online games, a playout buffer is not practical, so jitter does affect them, and it is in fact used as an input variable for some subjective quality models for First Person Shooters (Wattimena et al., 2006) or MMORPGs (Ries et al., 2008). For example, if the jitter is high enough to produce packet reordering, then it is translated into packet loss, since the game will not take into account e.g. a position update prior to the last one received. In some Real-Time Strategy games, as already said, commands are queued and executed about 200 milliseconds later due to the algorithm for state synchronization between all participating distributed clients. This could be considered as a sort of “playout buffer”.

Packet loss

Packet loss is another impairment which appears in networks. Different causes can prevent a packet from arriving to its destination in an IP network. Obviously, wireless networks have a higher packet loss rate than wired ones, although they implement different retransmission mechanisms at lower levels. Packet loss in wired networks is mainly caused by buffers: they may drop packets when they become full, but some policies (Active Queue Management Policies as e.g. Random Early Drop) may discard packets according to some statistical probabilities even if the buffer is not full.

For TCP-based games, packet loss is not directly considered as a network impairment, since this transport protocol will retransmit the lost packets. However, this is translated into an additional “spikes” of delay, since the packets arrived correctly will have to wait until the new copy of the lost packet arrives.

When the game is based on UDP, it is assumed that some packets may get lost. In VoIP robust codecs are used, able to reconstruct the original signal even in the absence of some of the packets. In online games, different techniques are employed so as to hide the effect of packet loss to the end user.

Many of the QoE-enhanced mechanisms are designed for hiding the effect of packet loss to the player. The client and the server may be able to do a prediction if a packet is lost, based on the current movement of the avatar. However, if the forecasted position and the one arrived in the next packet are not the same, then the player may observe a sudden and abrupt movement of the scenario. Although this can be mitigated by smoothing algorithms, it reduces the quality experienced by the player.

Different First Person Shooter games show very different degrees of robustness against packet loss. In (Zander and Armitage, 2004) two games were tested with real people, and the results regarding packet loss were very different: while *Halo* stopped working when packet loss was roughly 4 %, the users of *Quake III* did not experience any degradation in the quality even with a packet loss of about 35 %.

Bandwidth

Although bandwidth is not a network impairment by itself, its scarcity may be translated into high amounts of latency, jitter and packet loss. In addition, background traffic (i.e. bandwidth utilization) is also a parameter to be taken into account: a strongly congested network is not good for playing online games.

Cloud games require high amounts of bandwidth, especially in the downlink, so if the connection is not able to provide it, the quality of the video needs to be reduced (in terms of resolution or frames per second), thus reducing the subjective quality in turn. Other option is to suffer packet loss which severely degrades the quality of the video stream (blocking and blurring effects) and therefore of the game as well.

Developed QoE Models

As remarked in (Ubicom, 2005), different benchmarks exist for some computer applications:

- benchmarks for computer performance when running office applications are developed by certain companies (e.g. *PCMark* by *Futuremark*);
- the ITU’s E-Model can be used for estimating subjective quality in VoIP;
- video quality can be estimated by Peak Signal-to-Noise Ratio (PSNR) or with other estimators, called perceptual visual quality metrics (PVQMs), which may include user’s perception of video artifacts (Lin and Kuo, 2011);
- the main parameter for file-download applications is the throughput (or the time required for downloading the data).

However, as far as online games are concerned, it is very difficult to define a single benchmark able to summarize the quality experienced by the user. The quality of an online game may depend on two groups of system parameters: the hardware where the game runs, and the network. In some cases (i.e. game consoles) the hardware is fixed and games are specifically designed for it, so the only variable parameters (leaving apart the display and the speakers) are those related to the network. As said in (Ubicom, 2005), a benchmark able to summarize the effect of the network for online games should be able to:

- Capture the relationship between the objective metrics and the user’s satisfaction with the game;
- accurately represent the effect of the network and the other players;
- it should also be reproducible.

These QoE models should consider the network impairments having a real influence on the player’s experience with the game. For example, if the game has a very good method for concealing packet loss, this parameter may not be included, or included with a low influence.

These models are obtained using a cohort of volunteers who play the game while network impairments are added. After playing, they fill a survey about the experienced quality, and then regression techniques are employed to devise a mathematical model able to obtain a Mean Opinion Score, using as an input the value of each of the impairments. They usually consist of a polynomial formula, where each of the network impairments is weighted using a coefficient. Different statistical methods can then be used in order to estimate how well the model corresponds to the subjective quality results expressed by the users.

An example is the “impairment factor” proposed in (Ubicom, 2005):

$$\text{Impairment factor} = (W_L \times L + W_J \times J)\ (1 + E) \tag{1}$$

where W_L and W_J are the weighting factors for latency (L) and jitter (J) respectively, and E is the packet loss rate.

Other models obtain similar formulae for the impairment factor: in (Wattimena et al., 2006) only delay and jitter are considered when obtaining the impairment factor, taking into account that the considered game (a First Person Shooter) has a very good method for concealing packet loss.

$$\text{Impairment factor} = 0.104 \cdot \text{ping} + \text{jitter} \tag{2}$$

In (Ries et al., 2008) something similar happens: packet loss is not considered, taking into account that a TCP-based game is being tested, so lost packets are retransmitted. The formula is in this case:

$$\text{MOS} = 5.17 - 0.012 \cdot \text{delay} - 0.018 \cdot \text{jitter} \tag{3}$$

However, the models have many limitations:

- One model is only valid for a single title. Some generic models have been proposed, but the weighting factors have to be tuned according to each game, and new subjective tests are required.
- The models involve a simplification. For example, it has been demonstrated that the influence of a network impairment is modified depending on the activity performed by the player (Suznjevic et al., 2013). In this study, in which real players were involved, the impact of packet loss was stronger for the players fighting with other real ones in *dungeons* than for the ones *questing* (this activity is not normally performed in group).
- Most models do not take context and users parameters into account.

All in all, it can be said that the subjective quality estimation for online games is a much more complex problem than in the case of Voice over IP. Although some similarities can be found, a universally accepted formula for measuring it does not exist. This is sometimes translated into a simplification of the problem, just talking about latency and neglecting the effect of other network impairments as jitter or packet loss.

QoE-enhancing Mechanisms Employed in Game Support

Delay related methods

Client-side prediction

As remarked in (Bernier, 2001; Oliveira and Henderson, 2003), different methods can be employed to combat delay in online games. The so-called *client-side prediction* has been largely used in First Person Shooters. It can be divided into *input prediction* and *dead reckoning*, where input prediction hides the latency for the client-controlled actions while dead reckoning hides the latency of other participating players.

For input prediction we can consider this sequence: the client generates a player command, which is transmitted to the server, where the next status of the game is calculated and sent to all the clients, and each of them renders the scene for its player. However, during this interval, something has to be shown to the player. The method consists of performing the movement of the client locally, just assuming that the server will accept the command. The drawback of the method is that, if the server’s response does not fit with the client prediction, a perceptible shift in the position of the avatar will be appreciated by the player.

Dead reckoning is basically an algorithm for estimating the position of an entity in the virtual world based on its previous position, orientation, speed, acceleration and other parameters. For dead reckoning, after receipt of the first state protocol data unit (PDU) for an entity (e.g. the character of another player), each client in the virtual world net begins moving the entity by applying the agreed-upon dead reckoning algorithm. Its movement is refreshed by receiving subsequent PDUs. If a spike of delay or packet loss appears for packets carrying PDUs related to some of the entities, each copy of the virtual world will continue to show movement of those entities according to agreed dead reckoning algorithm until the next update has been received. In addition, when an inconsistency between the server status and the one predicted by the client is found, some games are able to make the transition to the new status less abrupt, using smoothing algorithms.

Although client-side prediction may cause inconsistencies, it is able to increase the QoE, since it hides network latency to the user. Thus, it is widely employed in online games, and some titles allow the user to tune its parameters (Bernier, 2001).

Server side delay compensation

In addition to client prediction, the server may perform a compensation method in order to correctly merge virtual realities which are out of sync due to network delay. The server stores the history of the recent player positions (e.g. servers running *Valve's Source* engine store players' positions for 1 second) and when it has to calculate the new state, it first estimates the moment when the action was performed in the client version of the virtual world state. In other words, the server "rewinds time" according to the particular client's latency, calculating execution of a particular command (e.g. whether the player's shot has managed to hit the target). The formula is:

Command Execution Time = Current Server Time - Packet Latency - Client View Interpolation

It uses the execution time of the command, the latency of the player and the movement prediction the client is using at that moment.

Fig. illustrates the case in which the client has 200 milliseconds of latency (RTT). It shows the screenshot taken on the server right after the server confirmed the hit. The red hitbox shows the target position on the client where it was 100ms + interplay period ago. While the player's command (i.e. shooting at the target) traveled to the server, the target continued to move to the left. After the user command arrived on the server, for the calculation whether or not that was a hit, the server restored the target position (blue hitbox) based on the estimated command execution time. The server traces the shot and confirms the hit.

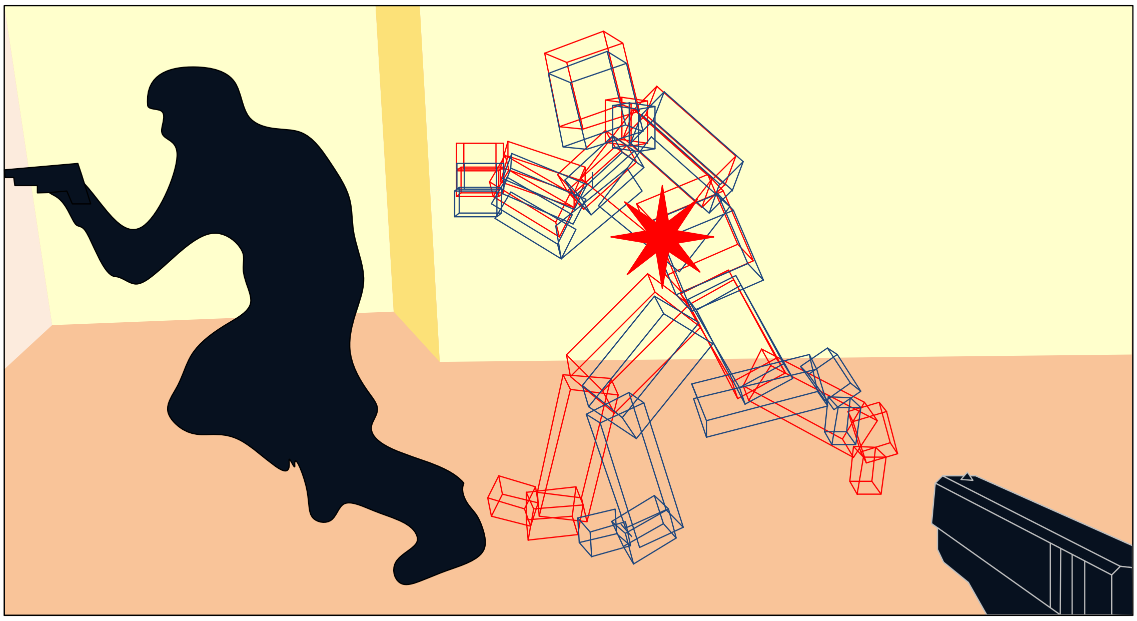


Fig. 9 Server side latency compensation

This sometimes can again result in anomalies as illustrated in Fig. 10 (on behalf of simplicity, client prediction is not considered): Jack's avatar is hidden between two buildings when Wang appears running forward. Jack makes a good shot towards Position 1 just when Wang is there. This shot is transmitted to the server. Simultaneously, Wang's computer transmits its new position to the server. When Jack's shot arrives to the server, it "goes back in time" just the time estimated as Jack's latency, and then it calculates if the shot has hit anyone. Wang was at Position 1 at that moment, so the server concludes that Wang's avatar has been hit, and transmits this information to both players. As a result, Wang may perceive that he has been shot "around the corner".

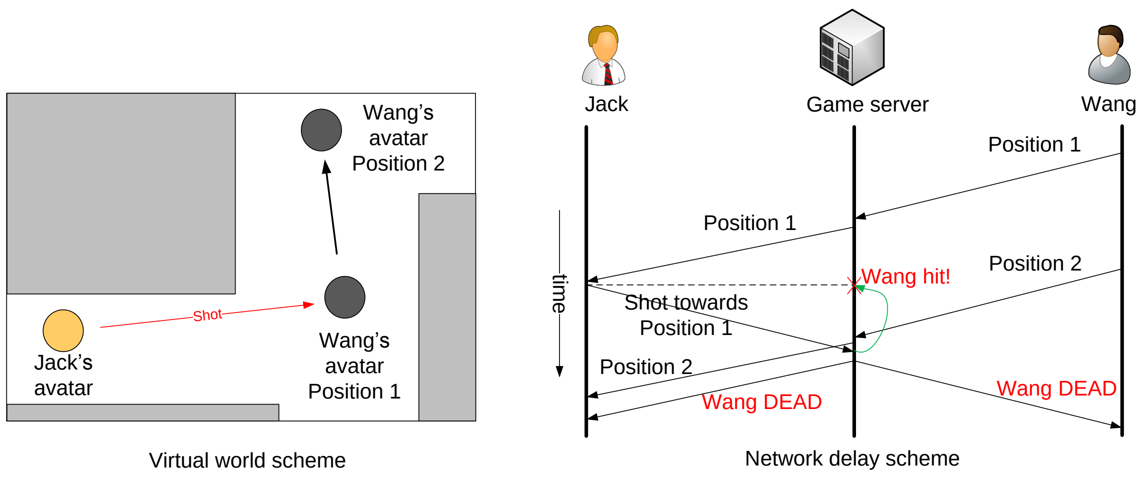


Fig. 10 Scheme of the server delay compensation mechanism

This anomaly is further illustrated in **¡Error! No se encuentra el origen de la referencia.**. In this case player being shot at (Wang) would be receiving a hit, when in his version of the virtual world he is already behind the cover, and it is physically impossible to get hit due to the cover provided by the wall. Wang may feel a bit annoyed with this apparently unfair behavior of the game.

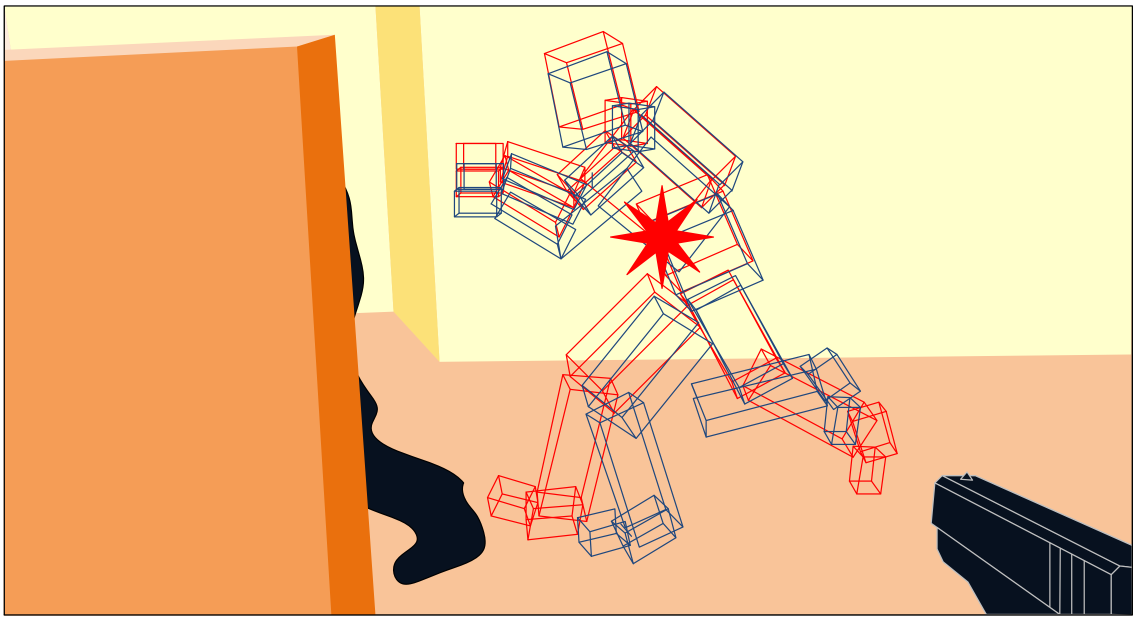


Fig. 11 Anomaly caused by sever side lag compensation

This problem could be solved if the client was able to send a "hit" message instead of a "shoot" one. However, servers cannot allow the clients to report hits, because of security reasons: some clients (or man-in-the-middle proxies) could then inject fake hits in order to increase the game performance of certain malicious players.

Geographical server distribution

A natural method for reducing latency is the optimization of server locations. The closer the server is to the player (physically) the transfer part of the latency is lower. Many games report the geographical place where their servers are, and they encourage the gamers to play in servers near their location. Sometimes this can be done automatically: the player enters in a queue, and an algorithm tries to match groups of players with similar skill levels and geographical locations, before starting a party between them. This happens in many First Person Shooters, where a party can be short and many players do not need to play against other specific people.

In other cases, players are allowed to select the server where they want to play. They may be able to select a server in another zone, but in that case they are aware of the possibility of having a higher latency.

Scalability related methods

In certain game genres (e.g. First Person Shooters or Real-Time Strategy), every client receives from the server information of the status of every avatar in the virtual world. This is good for consistency, since e.g. a map including the positions of the rest of the players (or only those in the same team) can be displayed. However, in this case, the amount of information to be sent by the server grows with the square of the number of players. Some studies have modeled the traffic depending on the number of players (Branch et al., 2008; Lang et al., 2005).

This scalability problem sets a limit on the number of players that can share a virtual scenario in these genres, and this problem persists: for example, *Counter Strike 1* (1999) was able to manage 32 players, and, twenty one years later, *World of Tanks'* (2010) normal parties involve 30 people.

However, one of the main attractive points of other genres (MMORPG) is its “Massiveness”. The fact of sharing a virtual world with a huge number of people is especially interesting for the players as it gives them the feeling of participation in a large virtual world with its economic and social specifics. Players may join a guild, participate in missions with other people, have a list of virtual friends, show the new weapons of their fancy avatars to lots of real people, etc. In fact, many of these games include thousands of people in the same virtual world. Thus, game developers implement different mechanisms in order to deal with this hard scalability problem.

Area of Interest

If all of the players in a massive virtual world received real time information regarding all of the other players, that would impose impossible requirements both on the network bandwidth and on the processing capabilities of the server. Therefore, an Area of Interest (AOI) can be defined around the avatar, also including its field of view. Then, only the information generated by other avatars in player’s AOI is sent to him/her. This allows the game developer to maintain consistency between all the players, and at the same time to keep the processing load and the traffic to be sent to the player in reasonable limits. This results in a statistical model of the traffic which depends on the number of players in the server: this effect was studied in (Suznjevic et al., 2014), where a model for an MMORPG *(World of Warcraft)* with a different statistical behavior depending on the number of players in a server was presented.

Sharding

A virtual world can be distributed across game servers in two ways: *a)* one logical instance of the virtual world is created for all players and is spanned across all the machines of the game server farm (e.g. *EvE Online*); *b)* the virtual world is replicated on more than one logical instance, called *shard*. Shards partition the player base across several logical instances of the virtual world (players in different shards cannot interact with each other), and through that the computational load is reduced. It should be noted that in recent times the limits of the shards in the most popular MMORPGs have been weakened. For example, in *World of Warcraft* (*WoW*) it is possible to communicate with players in other shards, or even other games, using Blizzard’s overlay communication system *battle.net*. Also, in *WoW* since recently it is possible for certain zones to be shared across different shards, so players from different shards are now able to play together.

Zoning

Scalability on a single shard is achieved through a parallelization technique called *zoning*. This technique partitions the virtual world into geographical areas called *zones*, which can be handled independently by separate machines, as depicted in Fig. 12. In the past, zones had borders (e.g. invisible walls) with transition spots between them, such as portals, because transition between zones required certain time. These events had a bad influence on the players’ immersion in the virtual world, which led to the development of the technique called *seamless zoning*, in which zones might be divided by some geographic markers (e.g. mountains), but there are no loading screens when players cross from one zone to another.

Mirroring

Another technique, called *mirroring*, targets parallelization of game sessions with a large density of players located and interacting within each other’s AOI. “Hot spots” appear in virtual worlds, and they typically include major cities or zones in which gathering of players is common. To address this problem, mirroring is performed by distributing the load by replicating the same game zone on several servers. In each replicated server, the state for a subset of entities (i.e. active entities) is calculated, while the remaining entities (i.e. shadowed entities) states are calculated in the other participating servers, and are synchronized across servers, as shown in Fig. 12. The overhead of synchronizing shadowed entities is much lower than the overhead of computing all entities as active entities (Muller et al., 2006).

Instancing

Instancing is a technique which can be perceived as a simplification of mirroring, or even sharding on a smaller scale. This technique distributes the session load onto multiple parallel instances for the highly populated zones. The instances are independent of each other, which means that two avatars from different instances will not be able to interact with each other, even if they are located at coordinates within their AOI. This technique is common in many MMORPGs. All scalability techniques, including instancing, are depicted in Fig. 12.

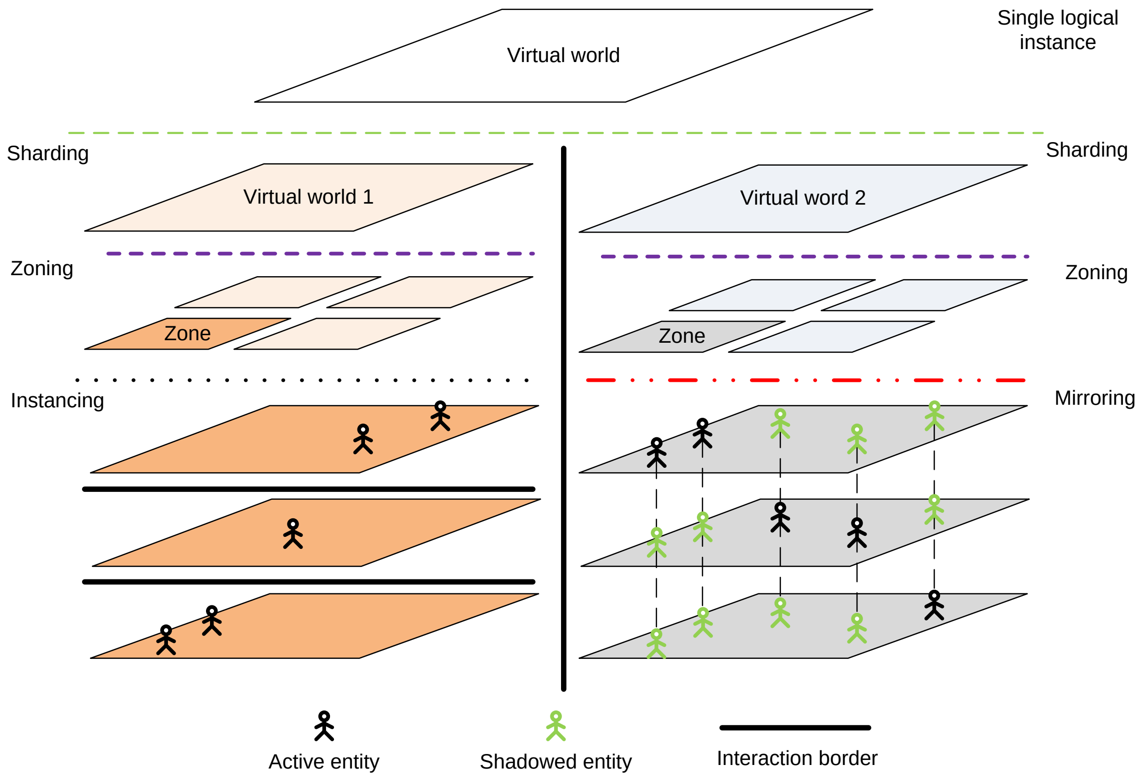


Fig. 12 Scalability techniques for online games

Conclusion

Companies employ different techniques to provide games in a scalable, reliable and profitable way, with the main objective of achieving a good Quality of Experience level. This concept not only includes Quality of Service, which is directly related to the parameters that can be measured from the network, but also captures the expectations, feelings, perceptions, cognition and satisfaction of the user. There is a special difficulty with this kind of service, since gamers show a very demanding profile. This chapter has discussed the different mechanisms that companies use in order to overcome the problem of network impairments (latency, packet loss, etc.) when providing online games.

The synchronization and the maintaining of a coherent game state to be shared by the applications of all the players is not a trivial problem: different sources of latency appear and may cause inconsistencies between the game states observed by each of the players. Different genres of online games present specific latency requirements, depending on the game dynamics, their characteristics and the level of interaction between the players. First Person Shooters are considered as the game genre with the tightest latency limits, but other genres also have interactivity requirements, so a "delay sensitivity" can be defined for each of them.

Online games have employed different architectures, being client-server the one employed in the vast majority of the cases. It has many advantages, as the existence of a central authority, cheating prevention, an easier implementation of billing functionalities, etc.

Online games (except those known as "cloud games") generate low-bandwidth traffic flows, in order to allow a high packet rate and constant updates of the client movements and the game status. Different protocols (UDP or TCP) are employed at Transport level, and this is somewhat related with the game genre and its interactivity requirements.

The different techniques used for estimating the user's Quality of Experience from network parameters always consider latency as the most important parameter. They obtain an impairment factor weighting latency with other impairments as packet loss, delay variation (jitter). Some factors may increase or reduce the effect of each parameter: for example, in TCP-based games, packet loss is not a problem by itself, since lost packets are retransmitted, but it is translated into an additional latency. Additionally, some First Person Shooter games are very robust against packet loss, but have more sensitivity to jitter.

Finally, different QoE-enhancing mechanisms are used by game providers, as client-side prediction, which try to hide latency not only of the local client, but also of the other participating players. Another mechanism is server delay compensation. When employed, the server goes back in time before calculating the results of a player action, taking into account his associated latency. Other scalability-related techniques are the use of an Area of Interest, which means that only the actions of the avatars near the user are sent to his client. Different methods are used in order to distribute the workload between different game servers, as creating multiple instances (*sharding*) of the virtual world, dividing it into zones, or parallelizing game sessions corresponding to densely populated zones.

Cross-references

Recommended Reading

Adams R (2013) Active Queue Management: A Survey. Communications Surveys & Tutorials, IEEE, vol. 15, no.3, pp.1425-1476, Third Quarter 2013. doi: 10.1109/SURV.2012.082212.00018

Batool SH, Mahmood K (2010) Entertainment, communication or academic use? A survey of Internet cafe users in Lahore, Pakistan. Information Development vol 26: 141-147. doi:10.1177/0266666910366650

Bernier, Y (2001) Latency Compensating Methods in Client/Server In-Game Protocol Design and Optimization. Proc. Game Developers Conference, San Jose, Vol. 98033. No. 425.

Branch PA, Cricenti AL, Armitage GJ (2008) An ARMA(1,1) prediction model of first person shooter game traffic. Proceedings IEEE 10th Workshop on Multimedia Signal Processing (MMSP '08): 736–741, Cairns, Australia. doi: 10.1109/MMSP.2008.4665172

Brun O, Bockstal C, Garcia J (2006) A Simple Formula for End-to-End Jitter Estimation in Packet-Switching Networks. Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies: 14,14. doi: 10.1109/ICNICONSMCL.2006.34

Cajada M (2012) VFC-RTS: Vector-Field Consistency para Real-Time-Strategy Multiplayer Games. Master of Science Disertation.

Available online: http://www.gsd.inesc-id.pt/~lveiga/prosopon/pubs/35-msc-cajada.pdf

Chambers C, Feng WC, Sahu S, Saha D (2005) Measurement-based characterization of a collection of on-line games. In Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement: 1-1. USENIX Association.

Chen K, Huang P, Chin-Luang L (2006) How sensitive are online gamers to network quality? Communications of the ACM 49, 11: 34-38. doi: 10.1145/1167838.1167859

Chen KT, Huang CY, Huang P, Lei CL (2006) An empirical evaluation of TCP performance in online games. Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE ’06), Hollywood, Calif, USA. doi: 10.1145/1178823.1178830

Chen J (2007) Flow in games (and everything else). Communications of the ACM 50.4: 31-34. doi: 10.1145/1232743.1232769

Claypool M (2005) The effect of latency on user performance in Real-Time Strategy games. Computer Networks, Volume 49, Issue 1, 15: 52-70. doi: 10.1016/j.comnet.2005.04.008

Claypool M, LaPoint D, Winslow J (2003) Network Analysis of Counter-strike and Starcraft. Proceedings of the 22nd IEEE International Performance, Computing, and Communications Conference (IPCCC), USA: 261-268. doi: 10.1109/PCCC.2003.1203707

Claypool M, Claypool K (2006) Latency and player actions in online games. Communications of the ACM 49, 11: 40-45. doi: 10.1145/1167838.1167860

Claypool M, Finkel D, Grant A, Solano M (2012) Thin to win? Network performance analysis of the OnLive thin client game system. IEEE 11th Annual Workshop on Network and Systems Support for Games (NetGames): 1-6.

Cole RG, Rosenbluth JH (2001) Voice over IP performance monitoring. SIGCOMM Comput. Commun. Rev. 31, 2: 9-24. doi: 10.1145/505666.505669

Debeauvais T, Nardi B (2010) A qualitative study of Ragnarök Online private servers: in-game sociological issues. Proceedings of the Fifth International Conference on the Foundations of Digital Games. ACM, 2010. doi: 10.1145/1822348.1822355

Deci EL, Ryan RM (1985) Intrinsic motivation and self-determination in human behavior (Perspectives in Social Psychology). Plenum Press, New York and London

Dick M, Wellnitz O, Wolf L (2005) Analysis of factors affecting players' performance and perception in multiplayer games. Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games: 1 -7. doi: 10.1145/1103599.1103624

Feng WC, Chang F, Feng W, Walpole J (2005) A traffic characterization of popular on-line games. IEEE/ACM Transactions on Networking 13.3: 488-500. doi: 10.1109/TNET.2005.850221

Ford M (2014) Workshop report: reducing internet latency, 2013. ACM SIGCOMM Computer Communication Review 44.2: 80-86. doi: 10.1145/2602204.2602218

Furuholt B, Kristiansen S, Wahid F (2009) Gaming or gaining? Comparing the use of Internet cafes in Indonesia and Tanzania. The International Information & Library Review, Volume 40, Issue 2: 129-139. doi: 10.1016/j.iilr.2008.02.001

Gautier L, Diot C (1998) Design and evaluation of mimaze a multi-player game on the internet. Proceedings IEEE International Conference on Multimedia Computing and Systems, 1998. doi: 10.1109/MMCS.1998.693647

Gettys J, Nichols K (2011) Bufferbloat: Dark Buffers in the Internet. Queue 9, 11, Page 40, 15 pages. doi: 10.1145/2063166.2071893

Gurol M, Sevindik T (2007) Profile of Internet Cafe users in Turkey. Telematics and Informatics, Vol 24, Issue 1: 59-68. doi: 10.1016/j.tele.2005.12.004.

Henderson T, Bhatti S (2003) Networked games: a QoS-sensitive application for QoS-insensitive users? Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS: What have we learned, why do we care?: 141-147. doi: 10.1145/944592.944601

Huang CY, Hsu CH, Chang YC, Chen KT (2013) GamingAnywhere: an open cloud gaming system. Proceedings of the 4th ACM Multimedia Systems Conference (MMSys '13). ACM, New York, NY, USA: 36-47. doi: 10.1145/2483977.2483981

Kaiser A, Maggiorini D, Boussetta K, Achir N (2009) On the Objective Evaluation of Real-Time Networked Games. Proc. IEEE Global Telecommunications Conference (GLOBECOM 2009). doi: 10.1109/GLOCOM.2009.5426032

Lang T, Branch P, Armitage G (2005) A synthetic traffic model for Quake3. Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE ’04): 233–238, Singapore. doi: 10.1145/1067343.1067373

Lee CS (2012) The Revolution of StarCraft Network Traffic. Proceedings of the 11th Annual Workshop on Network and Systems Support for Games NetGames 2012.

Available online: http://dl.acm.org/citation.cfm?id=2501583

Lin W, Jay Kuo CC (2011) Perceptual visual quality metrics: A survey. Journal of Visual Communication and Image Representation, Volume 22, Issue 4: 297-312. doi: 10.1016/j.jvcir.2011.01.005.

Miller JL, Crowcroft J (2010) The near-term feasibility of P2P MMOG's. IEEE 9th Annual Workshop on Network and Systems Support for Games (NetGames), Article 5, 6 pages.

Muller J, Metzen H, Ploss A, Schellmann M, Gorlatch S (2006) Rokkatan: Scaling an RTS game design to the massively multiplayer realm. Comput. Entertain. 4, 3, Article 11. doi: 10.1145/1146816.1146833

Oliveira M, Henderson T (2003) What online gamers really think of the Internet?. Proceedings of the 2nd workshop on Network and system support for games (NetGames '03). ACM, New York, NY, USA: 185-193. doi: 10.1145/963900.963918

Qualinet White Paper on Definitions of Quality of Experience (2012). European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), Patrick Le Callet, Sebastian Möller and Andrew Perkis, eds., Lausanne, Switzerland, Version 1.2, March 2013.

Ratti S, Hariri B, Shirmohammadi S (2010) A Survey of First-Person Shooter Gaming Traffic on the Internet. IEEE Internet Computing 14, 5: 60-69. doi: 10.1109/MIC.2010.57

Rehman-Laghari K, Crespi N, Molina B, Palau CE (2011) QoE Aware Service Delivery in Distributed Environment. Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on, vol., no., pp.837, 842, 22-25. doi: 10.1109/WAINA.2011.58

Ries M, Svoboda P, Rupp M (2008) Empirical Study of Subjective Quality for Massive Multiplayer Games. Proceedings of the 15th International Conference on Systems, Signals and Image Processing: 181 - 184. doi: 10.1109/IWSSIP.2008.4604397

Saldana J, Fernandez-Navajas J, Ruiz-Mas J, Wing D, Perumal MAM, Ramalho M, Camarillo G, Pascual F, Lopez DR, Nunez M, Florez D, Castell JA, de Cola T, Berioli M (2013) Emerging Real-time Services: Optimizing Traffic by Smart Cooperation in the Network. IEEE Communications Magazine, Vol. 51, n. 11: 127-136. doi: 10.1109/MCOM.2013.6658664

Suznjevic M, Dobrijevic O, Matijasevic M (2009) MMORPG Player Actions: Network Performance, Session Patterns and Latency Requirements Analysis. Multimedia Tools and Applications, vol. 45 no. 1-3: 191-214. doi: 10.1007/s11042-009-0300-1

Suznjevic M, Matijasevic M (2012) Player behavior and traffic characterization for MMORPGs: a survey. Multimedia Systems, vol. 19, no. 3: 199–220. doi: 10.1007/s00530-012-0270-4

Suznjevic M, Skorin-Kapov L, Matijasevic M (2013) The impact of user, system, and context factors on gaming QoE: A case study involving MMORPGs. Network and Systems Support for Games (NetGames), 12th Annual Workshop on. IEEE. doi: 10.1109/NetGames.2013.6820606

Suznjevic M, Saldana J, Matijasevic M, Fernandez-Navajas J, Ruiz-Mas J (2014) Analyzing the effect of TCP and server population on massively multiplayer games. International Journal of Computer Games Technology, vol. 2014, Article ID 602403, 17 pages. doi:10.1155/2014/602403

Svoboda P, Karner W, Rupp M. (2007) Traffic Analysis and Modeling for World of Warcraft. ICC '07. IEEE International Conference on Communications: 1612-1617. doi: 10.1109/ICC.2007.270

Tarng, PY, Chen KT, Huang P (2008) An analysis of WoW players' game hours. Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games. ACM. doi: 10.1145/1517494.1517504

Thompson B, Koren T, Wing D. (2005) RFC 4170: Tunneling Multiplexed Compressed RTP (TCRTP).

Ubicom White Paper (2005) OPScore, or Online Playability Score: A Metric for Playability of Online Games with Network Impairments.

Available online at http://www.kevingee.biz/wp-content/uploads/2011/04/IP3K-DWP-OPSCORE-10.pdf

Wattimena AF, Kooij RE, van Vugt JM, Ahmed OK (2006) Predicting the perceived quality of a first person shooter: the Quake IV G-model. In Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games (NetGames '06). ACM, New York, NY, USA, Article 42. doi: 10.1145/1230040.1230052

Wu CC, Chen KT, Chen CM, Huang P, Lei CL (2009) On the Challenge and Design of Transport Protocols for MMORPGs. Multimedia Tools and Applications, Vol. 45, No. 1: 7-32. doi: 10.1007/s11042-009-0297-5

Zander S, Armitage G (2004) Empirically Measuring the QoS Sensitivity of Interactive Online Game Players. Proc. Australian Telecommunications Networks & Applications Conference (ATNAC 2004), Sydney, Australia, Dec. 2004.

Keywords

Online games, Quality of Experience, Latency, Network Impairments, Real-time interaction