Analyzing Unreal Tournament 2004 Network Traffic Characteristics[1]

Christian Hübsch
University of Karlsruhe, Institute of Telematics
Zirkel 2, 76128 Karlsruhe, Germany
+49 721 608 6405
huebsch@tm.uka.de

Abstract

With increasing availability of high-speed access links in the private sector, online real-time gaming has become a major and still growing segment in terms of market and network impact today. One of the most popular games is Unreal Tournament 2004, a fast-paced action game that still ranks within the top 10 of the most-played multiplayer Internet-games, according to GameSpy [1].

Besides high demands in terms of graphical computation, games like Unreal also impose hard requirements regarding network packet delay and jitter, for small deterioration in these conditions influences gameplay recognizably. To make matters worse, such games generate a very specific network traffic with strong requirements in terms of data delivery. In this paper, we analyze the network traffic characteristics of Unreal Tournament 2004. The experiments include different aspects like variation of map sizes, player count, player behavior as well as hardware and game-specific configuration. We show how different operating systems influence network behavior of the game. Our work gives a promising picture of how the specific real-time game behaves in terms of network impact and may be used as a basis e.g. for the development of specialized traffic generators.

Keywords

Network Traffic, Real-time Games

1. Introduction

Since the Internet has advanced to be the most widespread and accessible medium for global communication, games always played an important role in its profile of usage. In many parts of the world today it is normal to have access to a comparably fast Internet connection, enabling to take part in such games. Several classes can be differed, ranging from turn-based or slow-paced interactive games to fast-paced real-time representatives. Today, the most relevant game types in terms of supporters are the Massively Multiplayer Online Games and First Person Shooters. Both game types proceed in real-time, although they vary in their requirements for fast interaction. While Massively Multiplayer Online Games rather tolerate small delays during gameplay, First Person Shooters often demand for immediate player reactions (requiring network delays below 100 ms [7]). The high popularity of such games leads to a notable fraction of global Internet traffic, typically characterized by traffic patterns of many small packets in short intervals. Since data delivery in today's Internet is based on a best effort service, it is desirable for providers to understand such traffic in order to improve the network to meet customer's demands.

This paper focuses on the network traffic characteristics of Unreal Tournament 2004 (UT), a popular First Person Shooter that ranges between the most-played members of that class [1]. By analyzing

[1] This work is part of the SpoVNet project supported by the Landesstiftung Baden-Württemberg

the traffic generated in various game sessions we characterize the network impact of UT. We consider the number of packets, the inter packet times and the payload sizes. It is shown how these measures depend on the number of players, the player behavior, and in-game interactions. Furthermore, we show that CPU and graphic adapter performance as well as operating system have large impact on the network traffic, whereas other factors like main memory have almost no impact.

Games like UT are based on game engines that support, e.g., graphical features, scripting, and network communication. The UT engine is closed source and licensed to game developers for a high fee. Thus, its inner organization and its communication protocols are not known so that conclusions about the engine's behavior can only be drawn by observing the network traffic. However, we point out that even knowing the characteristics and protocol behavior of an engine does not lead to a complete understanding of the network traffic that significantly depends on the game's dynamics.

This paper is organized as follows. In Section 2 we introduce UT and give an overview of the game. In Section 3 we briefly describe how network traffic data was obtained during the experiments. Section 4 describes results concerning important traffic aspects. Finally, concluding remarks are given.

2. Unreal Tournament 2004

Unreal Tournament 2004 [2], released by Epic Games [3] and Digital Extremes [4] in March 2004, is the third descendant of the Unreal Tournament branch. It is sometimes viewed only as an extension to its predecessor Unreal Tournament 2003 but was and is able to find an immense number of players. While Unreal Tournament 2003 uses the Unreal Engine 2, released in 2000, UT builds upon the Unreal Engine 2.5, improved in rendering and supporting 64-Bit architectures as the first closed-source game engine. As a typical representative of First Person Shooter games, UT provides a wide range of game modes differing in target and proceeding, but in most cases the game is extremely fast-paced and intense. In this paper, we analyze the Deathmatch game mode, being the most interactive mode (as all players are opponents in the virtual arena). This game mode represents the upper bound of network impact as it creates the highest amount of network game traffic.

Furthermore, concerning the official UT statistics [5], Deathmatch is still the most-played game mode. While Massively Multiplayer Online Games are dedicated to support extremely high numbers of players, First Person Shooter games tend to limit game sessions to player counts of several dozens in the maximum, depending on the particular game mode. In Deathmatch games, the number of players normally resides below 20 (even below 10 in most cases). Specifically to this game mode, various maps of different sizes are available, representing virtual arenas. UT has been published for Windows, Mac OS X and Linux.

3. Experimental Setup

For obtaining network game traffic, different numbers of client machines were connected to a server through a GBit LAN. Those machines varied in hardware specification and operating system, building a heterogeneous group of participants. The server machine ran a dedicated UT server software, not participating in active gameplay. All machines gathered network traffic to and from the server locally by running wireshark [6], a network packet analyzing tool. Participating players varied in skill and

experience and were allowed to set the game settings at will. All game entities were at the latest game patch level (3369). To gain representative in-game network data, numerous game sessions of at least 10 minutes duration were accomplished. The captured traffic was truncated to session parts in which all clients already joined the game, additionally cutting off the first game minute (mostly influenced by a short orientation and e.g. a weapon search phase) as well as the last 10 seconds of each session. Overall, we examined over 130 game sessions and 70 Million game packets.

4. Traffic Characteristics

In this section we analyze the characteristics of UT in-game traffic by analyzing different representative factors obtained during several Deathmatch game sessions. In Sec. 4.1 we give statements about some general observations we made concerning the game. Sec. 4.2 to 4.4 describe the rate of generated packets, their temporal offsets, their data sizes and the resulting bandwidth consumptions, respectively. After that, in Sec. 4.5 and 4.6 we observe the influence of different client hardware specifications and player in-game behavior.

4.1. General Observations

During the experiments, several general observations were made that should be considered as meaningful for evaluating UT game traffic. First of all, the engine's network cycle (determining when packets are emitted) is closely coupled to its rendering cycle (determining the calculation of visual frames in the game). While future real-time games may employ different threads for those tasks, each assigned e.g. to different CPUs in multi-core architectures, this is not the case in UT, as it is in most existing First Person Shooters. This leads to a high variance in client network behavior (described in the following sections). Also, some client machines (e.g. laptops) tend to overheat due to insufficient cooling capabilities at phases of high game load during the experiments. The CPU processing speed was cut down to half automatically in these phases, influencing the game's rate of frames per second. As the target of the experiments was to analyze representative UT game traffic, we do not ignore such cases but limit ourselves to point them out, for they may also occur in real Internet gaming as well.

4.2. Packets per Second (PPS)

During the game, data is exchanged between server and clients at specific rates per second, the Packets per Second rates (PPS). Clients send their game updates to the server, which has to aggregate them and disseminate them among all players. In this paper, we refer to PPS as the number of packets that has been sent during one second of game time, thus arranging those packets in one time bucket. The PPS differs in both directions and thus has to be analyzed separately. The oberservations show that the PPS from server to clients depends on how much information has to be published at the very moment as the game proceeds. If no player update information is needed (i.e. if the player took no action), the server constantly sends 6-8 packets per second to each client, as shown in Fig. 1. Here, 5 players joined a game without moving over the whole game time. This emitted traffic can be seen as a lower bound in UT PPS generation. As soon as player movement

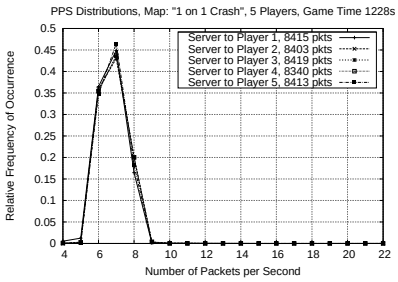

Fig. 1: PPS Server to Clients Distributions, 5 Players, all idle

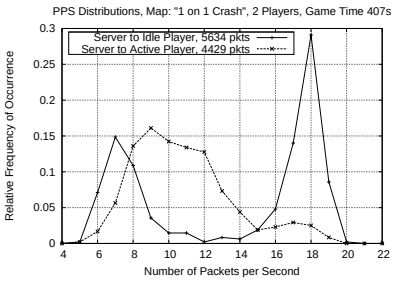

Fig. 2: PPS Server to Clients Distributions, 2 Players, one active

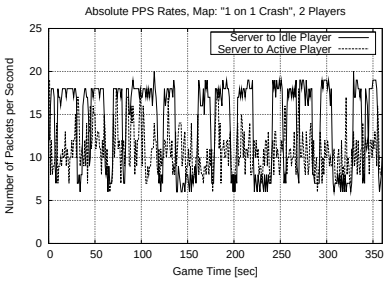

Fig. 3: PPS Server to Clients, 2 Players, one active

information has to be disseminated, being the normal case in active gameplay, the PPS from server to clients increases. Fig. 2 shows the rates in a game where one player did not move while the other one was acting representatively for normal player behavior. The PPS rate from the server to the idle player consists of two main peaks, one of which also covers the range from 6-8 packets and one around 18 packets. As the active player moves near the idle one (inside its 'area of interest', AOI), the latter has to be informed about its actions. This is accomplished by inceasing the PPS to the client to be informed, providing shorter delays between updates to ensure client-side "up to date" game states, a crucial factor for a good gaming experience. In game phases where no status update information has to be provided (e.g. because both players are located outside their mutual AOI), the PPS rate covers the same range as in idle times (compare Fig. 1). Active players also obtain more packets from the server than when idle, but less than their opponents. This is due to the fact that only a part of their actions has to be acknowledged by the server (like shooting), while e.g. movement information may be decided autonomously unless there is a conflict with the server state. Therefore, player actions are sent back by the server in PPS rates that vary from 6 to 19 packets (with 10 packets in the mean). While Fig. 2 shows the PPS distributions, Fig. 3 illustrates the absolute rates for the same case, showing how the idle player alternates between phases with the opponent inside and outside the AOI, respectively. The active player's PPS varies stronger over time, reflecting own player actions. Obviously, as more players take part in a game session, the overall interaction between players increases because it is more likely for them to get into one another's AOI in the virtual world. Since map sizes in the observed Deathmatch game mode range from very small to moderate-sized, the probability of interaction increases relatively fast, impacting network traffic. This effect is shown in Fig. 4, where the probability distribution functions of server to client PPS are displayed for a comparably

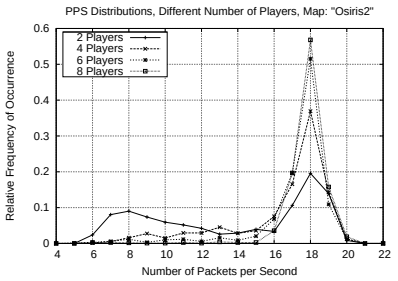

Fig. 4: PPS Server to Clients Distributions, Different Players N

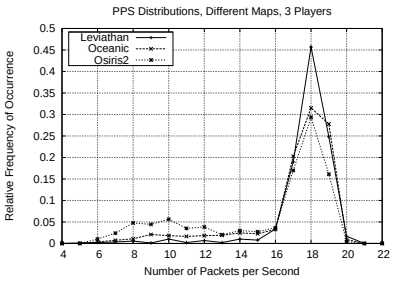

Fig. 5: PPS Server to 3 Clients Distributions, Different-sized Maps

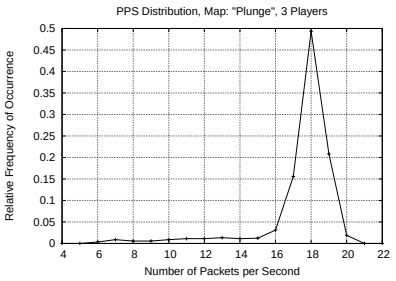

Fig. 6: PPS Server to Client, Plunge, 3 Players

large map (‘Osiris2’) with different player numbers. With more players taking part, the fraction of 18 packets per second grows, while lower fractions decrease. Similar results are shown in Fig. 5, displaying PPS rates for three different-sized maps, but the same player count, respectively. In game session on the map ‘Leviathan’ there are nearly no idle phases, while on ‘Osiris2’ more moments without contact occur. The grade of interaction is not assessable offhand only by the broadness of a map and the number of players, though. Crucial is only the probability of virtual contact. This becomes clear when observing ‘Plunge’, a large map (in terms of its dimensions) which nearly in any point of game time allows seeing one’s opponents due to its structure (compare Fig. 6). Here, the PPS is located near 18 even with few players.

As the PPS from server to clients follows the stated behavior, PPS from a specific client to the server is more complex to determine. Analysis shows that the rate of packets emitted by a client is directly coupled to its rate of processed frames per second (FPS). This leads to high fluctuations, depending on the client machine’s rendering capabilities, as well as high differences between different clients. Fig. 7 shows the PPS for a client that switches its rendering resolution during gameplay, affecting its PPS linearly to these changes. Analysis has shown that the maximum FPS rate is limited to a certain value (90 in Fig. 7) when playing UT over a network (also shown in Fig. 7), playing locally does not limit the reachable FPS rate. Furthermore, the engine aims for emitting one client packet per calculated frame, unless a specific FPS border is exceeded. This border has been delimited to be located between 45 and 55 FPS. As soon as the rendering cycle exceeds this value, the engine starts switching to sending a packet only every second frame. This is reasonable to reduce network load, due to missing advantage of higher PPS rates. Fig. 8 shows the PPS of 3 selected clients during a multiplayer game session. It has to be pointed out that the agility of a player (in terms of actions per time) has no influence on its PPS, crucial is the rendering performance. This is due to the fact that in most real-time games today, the rendering cycle is still connected to the network cycle.

4.3. Inter-Packet Times (IPT)

Packets sent to or from the server follow specific temporal offsets between one another, described through the Inter-Packet Times (IPT). While Sec. 4.2 examined packets in time buckets, this is not exact enough to draw conclusion about their temporal offsets. The delays between two consecutive data packets sent to the server by a specific client depend on the client’s current PPS rate. As the PPS is determined by the client’s FPS (as shown in Sec. 4.2), the IPT fluctuates following the

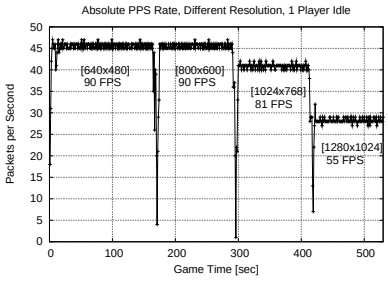

Fig. 7: PPS Development at Different Resolutions

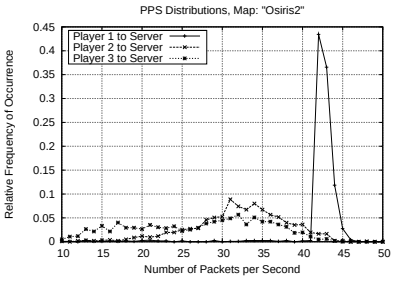

Fig. 8: PPS Different Clients to Server Distr., Osiris2

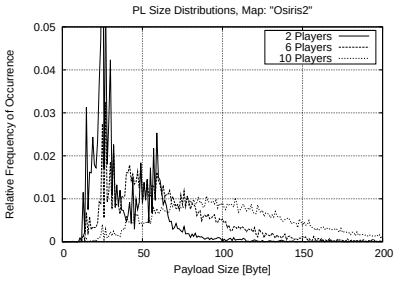

Fig. 9: PL Sizes Server to Client

particular client's characteristics, game situation and rendering cycles. Given a current PPS, the IPT arises from the equally temporal offset distribution between the packets. The IPTs from server to clients in contrast follow some observable regularities. Analysis of a game session with idle players showes that despite the absence of game interaction, the server sends two packets to each player periodically every 280 ms. The second of the two occurres exactly 56 ms after the first. For those two packets have different data sizes (which stay constant in each period), we conclude that UT uses two packet types for updating the clients local game states. This behavior fits well with the observed PPS rates (Sec. 4.2), as in idle phases the server sends 7 packets per second in average. In game sessions with more interactivity, the IPTs converge against constant offsets of 56 ms due to the rate of 18 packets per second to each client. In conclusion, the server decides in periods of 56 ms if data has to be disseminated to clients. In case game state information has to be sent, the server transmits packets to the relevant clients or otherwise remains without transmission.

4.4. Game Payload and Bandwidth Consumption

Payload size (PL) (the amount of application data transmitted in a specific data packet) in UT depends on the amount of data that has to be transmitted at a specific point of time. As mentioned in Sec. 4.3, PL size from server to clients in 'idle' game sessions shows peaks of occurrence in 2 main classes (around 20 and 45 bytes, respectively). We consider these packets to carry periodic status updates that occur periodically, even without further interaction in the game. The amount of data the server has to disseminate to each client naturally depends on the quantity of events that has to be sent to each specific client and therefore grows in average with the number of active players. Fig. 9 shows the PL size distribution from server to one client in various game sessions with different player numbers. It is clear that as more players create more information to be disseminated by the server, the PL sizes grow by the additivity of game events. Also, the PL sizes sent to a specific client closely depend on the amount of payload this specific client previously transmitted to the server (due to the fact that information gets acknowledged by the latter). In contrast to this development, PL sizes generated by a specific client partly depend on its currently emitted PPS rate, the player behavior (see Sec. 4.6) and the level of interaction. Fig. 10 shows three different PL size distributions, one that represents an idle client, one for a client involved in high interaction and one in a game session with low level of interaction. It can be observed here that idle clients nearly constantly send packets of PL size around 20 bytes to the server. As clients act actively and agile, the PL sizes increase, being partitioned into several dedicated classes. This is due to the fact that the client sends its current actions to the server, resulting in packets that carry information over none, one or more actions (and therefore exhibiting dedicated sizes). Resulting from that, clients in less interactive environments posess PL size distributions with a higher fraction around 20 bytes, representing game phases with less player agility. Additionally it should be mentioned that agile clients with relatively low PPS rates sometimes need to send more actions in one packet, resulting in occurrences of higher PL sizes at times. While PL sizes only consider application data, bandwidth (BW) consumption is determined through the combination of communication protocol headers (UDP, IP) and PL. As such, the bandwidth consumed by a UT server in a specific range of time is the sum of data sent to all clients

within this period. BW consumption at the server link typically increases by around 20 Kbits/sec per client. The more interaction takes place in the game session, the more the bandwith consumption will grow (until the upper bound is reached in terms of player agility). E.g. comparing a session with 2 players to a session with 3 players makes a difference of 16 Kbits in average, while comparing the latter with a 4 player session increases the BW consumption by 20 Kbits. The BW consumed by a client is determined through its PPS rate and its generated PL sizes, typically ranging between 16 to 23 Kbits/sec in game sessions with little interaction and 18 to 25 Kbits/sec in highly interactive games or for clients that reside at the upper bound of the engine, concerning its FPS.

4.5. Hardware and Operating System Impact

For hardware performance impact conclusions, we limit ourselves to client machines (assuming that the server is chosen to fit its dedicated role). As stated in Sec. 4.2, client PPS rates highly depend on its current FPS rates. This factor also influences IPT, PL and BW as has been shown. As such, rendering performance (determined by the graphics card and the CPU to some extent) is the crucial factor when observing hardware impact, other aspects (e.g. main memory) did not affect the network traffic notably during the experiments. It should be mentioned that FPS may not only be increased by better hardware configuration but also e.g. by playing at lower resolutions or downgrading the game's graphical effects (some players in the experiments chose to do so). Besides higher FPS, less fluctuations of all observed factors occurred, because the game engine is able to hold its FPS rates, even more if the FPS rate is located at the specific client's FPS limit and thus rather limited by local constraints than by computational disability (compare Sec. 4.2). As also stated in Sec. 4.2, clients that remain narrow to the FPS rate at which the engine generates one packet per frame impact the network most in terms of packet count. Concerning the employed operating systems (OS), the experiments showed that all OSs behave similar when running UT, but there are some factors to point out. It could be observed that when running Linux, the maximum allowed FPS rate during game play was lower than when running e.g. Windows. Also, some Linux client machines showed indeterministic engine behavior, independently of how the player was acting. In these cases, the engine switched the game's FPS rate between several classes every couple of minutes, affecting most factors of the client's network impact. On the same machines this effect could not be detected when running Windows. Most probably these incidents may be traced back to graphics card drivers or similar reasons we were not able to figure out, but they are Linux-specific, as the analysis showed.

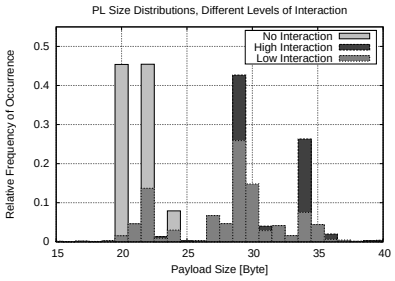

Fig. 10: PL Distributions Clients to Server, Different Interaction Level

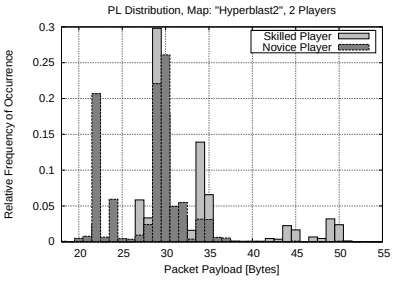

Fig. 11: PL Distributions Clients to Server, Different Player Skills

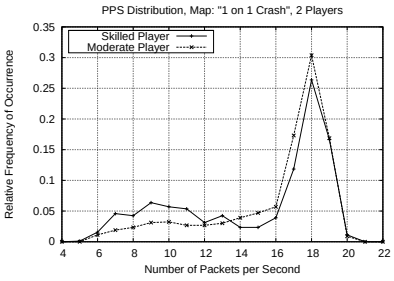

Fig. 12: PPS Server to Clients Distributions, Different Player Skills

4.6. Player Behavior and Skill

The characteristic (virtual) behavior of a UT player influences the generated network traffic in many respects. A player could choose to wait in a specific place to ambush his opponents (often called 'Camping') or run through the map highly agile, assaulting anything that moves. Those two different playing types of course come with different generated network traffic. The experiments have shown that highly skilled players generate partially higher PL sizes (due to the amount of actions they accomplish in parallel). Fig. 11 shows the difference in PL size distribution for two players in the same game session that differ in their skills. Also, as skilled players act more agile (unless they are campers) and generate more game state information, they arrange for other players to obtain more packets. Fig. 12 shows the compared PPS distributions for two (different skilled) players, revealing that the skilled player receives less packets as a result of his opponent acting less agile. Thus, by knowing a client's hardware specification, conclusions could be drawn on its behavior by only analyzing the network traffic.

5. Related Work

Several works exist examining network traffic of specific real-time games, e.g. [7] [8] [9] [10]. Those works mainly focus on how to model the network traffic of each specific game, while we also draw conclusions about how the engine behaves depending on various factors.

6. Conclusion and Future Work

This paper presented an analysis of characteristic network traffic generated by Unreal Tournament 2004, a popular real-time First Person Shooter game. Considering the number of packets generated per second, the inter-packet time and the payload sizes, we showed how the number of players, the player behavior and in-game interactions affect game traffic. Whereas the traffic from the client to the server roughly depends on the rate of processed frames per seconds, server to client traffic increases for active players as well as for idle players when an active player is moving in their area of interest. As additional factors we considered hardware and operating system specification. We found that the number of frames per second that can be processed by CPU and graphic adapter linearly determines network traffic until an upper bound is reached. As differences between Linux and Windows we found that the maximal number of frames per second is lower in Linux. In the future, the work may be part of a specific UT traffic generator or a more generic generator for the class of real-time games.

References

[1] *Game Spy Web Site.* http://archive.gamespy.com/stats/. March 2008
[2] *Unreal Tournament 2004 Web Site.* http://www.unrealtournament2003.com/ut2004/. March 2008
[3] *Epic Games Web Site.* http://www.epicgames.com. March 2008
[4] *Digital Extremes Web Site.* http://www.digitalextremes.com. March 2008
[5] *Official Unreal Tournament 2004 Statistics Web Site.* http://ut2004stats.epicgames.com/. March 2008
[6] *Wireshark Web Site.* http://www.wireshark.org/. March 2008
[7] Färber, J. Traffic Modelling for Fast Action Network Games
Multimedia Tools Appl., Kluwer Academic Publishers, 2004, 23, 31-46
[8] Lang et al. A synthetic traffic model for Half-Life, 2003
[9] Lang et al. A synthetic traffic model for Quake3, ACE '04: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology, ACM, *2004, 233-238*
[10] Färber, J. Network game traffic modelling,
NetGames 02: Proceedings of the 1st workshop on Network and system support for games, ACM Press, 2002, 53-57