Proceedings of the Ninth IASTED International Conference
INTERNET AND MULTIMEDIA SYSTEMS AND APPLICATIONS
February 21-23, 2005, Grindelwald, Switzerland

ONLINE GAMING MODELS FOR WIRELESS NETWORKS

Philipp Svoboda, Markus Rupp
TU-Wien
Institut für Nachrichten- und Hochfrequenztechnik
A-1040 Gußhausstraße 25/389
Vienna, Austria
psvoboda@nt.tuwien.ac.at

ABSTRACT:

In this paper, we present a possible extension of online gaming models with respect to wireless interfaces, such as UMTS. The first add-on is to generate a user-based model to analyse the typical length of a session and simulate traffic steps which will be critical to networks with shared resources. Next, mean opinion score (MOS) figures were adapted to wireless needs. In wireless channels, delay will most likely occur in combination with packet loss, therefore a 3D MOS figure is proposed. Such a 3D-MOS Figure facilitates the search for an optimal operational point given special construction of the wireless networks.

KEY WORDS

Networked Multiplayer Games, User-Model, 3D-MOS, Wireless

1. INTRODUCTION

In the past years online games have established themselves as a fixed part of the traffic shares in internet backbones. As this kind of service has completely different requirements than the "classical" services other strategies are required to optimize performance.

This year commercial UMTS started in Austria. According to the specifications this technology can provide service parameters sufficient for gaming. So our initial approach was to install an UMTS datacard and try to play Star-Craft and Unreal Tournament. Star-Craft worked out quite well but Unreal Tournament 99 (UT99) was unplayable due to high jitter. For us this was unexpected, because UT99 is an old game with bandwidth needs lower than 64kbit/s. To analyse the gaming in wireless networks we first measured and analysed the structure of gaming traffic in ideal environments. To evaluate future outputs of network simulation tools we investigated the influence of delay, jitter, packet loss and limited bandwidth on the player´s satisfaction (fig.1).

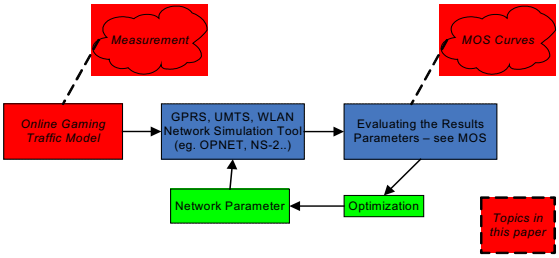

Figure 1: Defining the goal

Preliminary work concerning network characteristics of special games such as *Quake2* [1] [9], *Halo*[3] and lately *Unreal Tournament 2003* [4] and *War-Craft 3* [8] has been performed. The work is mostly focused on building accurate traffic models for performance evaluation of the underlying network. Our focus is however different. In wireless systems parameters like delay, packet loss or bandwidth are linked together via parameters like link power and so on.

In this paper we start with a classic analysis of the traffic for *Unreal Tournament* and *Star-Craft*. The two games were chosen due to our experience on both. Thereafter we give a short overview to Mean Opinion Score (MOS) ratings for both games. In particular, Section 5 is judging delay versus packet loss. This can be seen as a first step for further cost function optimization. To ease up this process, Section 4 analyses both games with respect to the user level. The user model for the first person shooter (FPS) is especially challenging.

Section 2 will show the measurement setup, where a FreeBSD [2] bridge with a dummynet implementation was used.

2. SETUP

2.1. The Games

Unreal Tournament 99 (UT99) [10] is a game quite popular in the world of FPS. In other words: the games are viewed in a first person perspective, and the goal is: shooting opponents. An important factor in this type of games is timing and accurate movement, making it challenging to set up on wireless systems.

Star-Craft (SC) [11] is one of the most popular real-time strategy (RTS) games. There are three tasks to master in this kind of games: First collect resources (often two types), second, use them wisely to build units and third, finally go to combat. Timing in these games is not that crucial, but higher delays also result in problems for micro management of the built units [8].

2.2. The Staff

It should be mentioned that we could only use a small "test sample" of five persons. However, the group was nicely balanced, consisting of four good players and one expert, a so called pro-gamer, for both UT and SC.

2.3. The Hardware

The setup is shown in Figure 2. One PC was separated from the rest using a FreeBSD machine with the dummynet firewall extension. The dummynet implementation in FreeBSD features delay, bandwidth and packet loss settings per network interface and service. The original granularity of the system timer is set to 10ms, introducing a random jitter. While in normal firewall applications a jitter of 10ms will be tolerated, this is critical to FPS games. Therefore, the FreeBSD system timer was recompiled with a new time step of 1ms. A validation of the setup , as pictured in Figure 2, showed a delay variation of about ±2ms from the set value (Fig. 2).

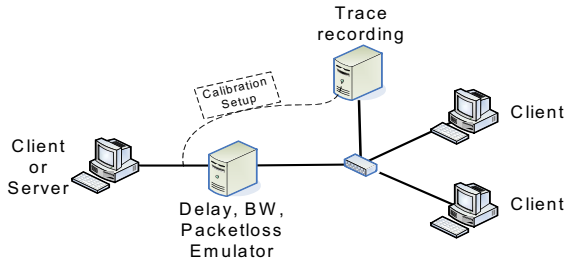

Figure 2: Calibrating and separating one node

Packet loss in dummynet can only be modelled as a uniform distribution, quite unlikely to occur in a real-world situation.

3. PACKET LEVEL ANALYSIS

All of the following graphs and analyses were performed without the dummynet system being activated to obtain the traffic characteristic of the games.

3.1. Real-Time Strategy (Star-Craft)

The packet size was shown to be constant at around 95 Bytes. The packet rate in LAN games is also constant (30ms). This is probably due to the age of the game and the missing CPU power to perform traffic shaping. The packet rate itself is also constant, this can the explained by the nature of the game. The game will run on all clients synchronously. The network is treated as an external input for the enemy forces. The packet stream represents periodical updates of the other player's mouse-clicks (everything else is executed on each client separately).

Introducing serious packet loss to such connections, most likely kills the session instantaneously. It is also worth mentioning that the traffic is growing linearly with the number of players, as each client has to send its packets to every other client on the network.

In Section 4.1 we analyze the session time distribution.

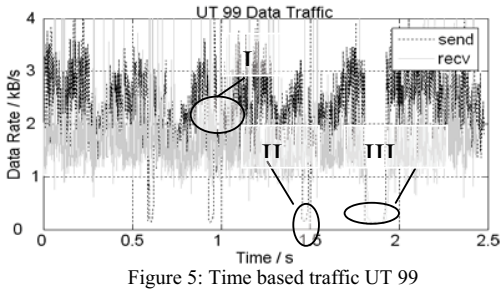

Figure 5: Time based traffic UT 99

3.2. First Person Shooter (Unreal Tournament 99)

We preferred UT99 instead of UT 2003 or UT 2004 for two reasons. First, the quite old UT99 still holds a large fan community. Second, short measuring traces for UT2003 showed that the traffic is similar. Also there is a strong tendency to up-scale bandwidth restrictions to at least ISDN or xDSL levels. See [3] for more details.

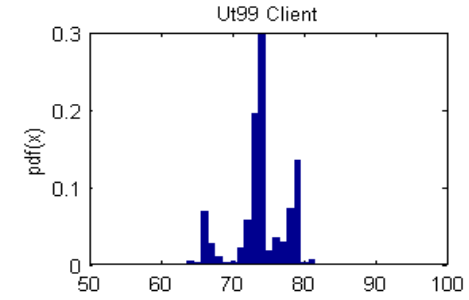

Figure 3a: Packet Size / Bytes

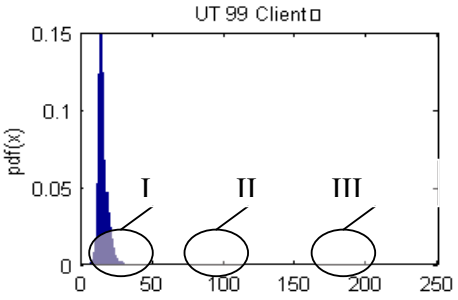

Figure 3b: Packet Interarrival Time / ms

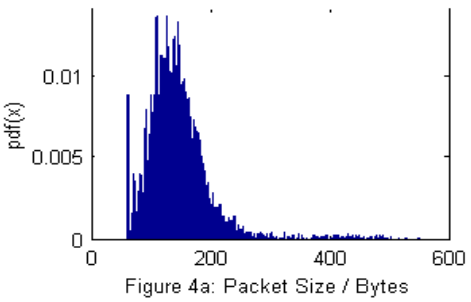

Figure 4a: Packet Size / Bytes

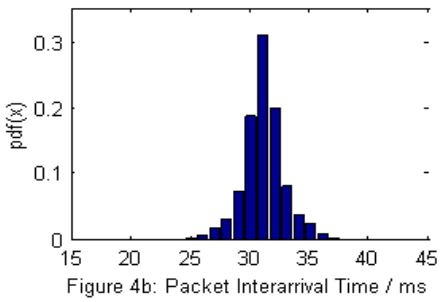

Figure 4b: Packet Interarrival Time / ms

The server is using a constant packet rate (Fig. 4b) to update the client constantly with a variable packet size (Fig. 4a). The client is using a constant packet size (Fig. 3b) with a variable rate (Fig. 3a). Of special interest is the client packet rate as indicated by the three circles. The small packet rate represents a traffic decrease caused by the player getting killed or if a level is changed. Therefore, we adopt our model to the user specific behaviour – as frag rates (a frag is a killcount; one gets a frag if one kills "frags" another player) [12] and typical session length (time a level is played). In addition there is the question if the up- and down-rates are to be correlated. Visual inspections showed that the down-traffic increases if the activity of the other players rises, in the same way as the upload traffic rises if the player on the client side is initiating more actions. Thus it is possible that the client up-traffic follows the down-traffic with some time delay, as a reaction to the increasing action in the game. More details can be found in [5].

4. A USER MODEL

4.1. Star-Craft

As shown in Section 3.1, the basic network traffic of SC is quite simple to reproduce. The only parameter left to obtain for a user model is the game time. Therefore, we extracted the session times from approximately 500 games

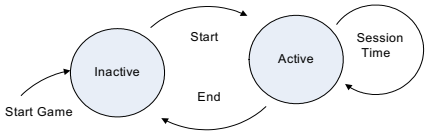

Figure 6: State diagram for a SC user model

we played during the last four years. We reduced the set to 432 by neglecting “fun” games.

Figure 7 shows that the time a game lasts exhibits a few particularities. First there is a restriction to the time a

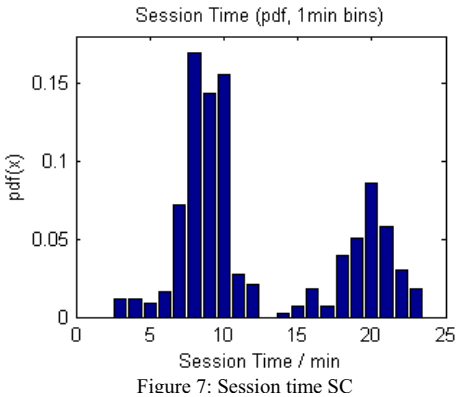

Figure 7: Session time SC

party needs to build up units, therefore there are no games lasting under 2min. After 2min so called rushes (quick and risky attacks) are possible, but quite rare. The second particularity which occurs at around 8min, corresponds to the time a player needs to build a small army and to ambush the opponent (on big maps the movement will also add some time). If one player is successful, the game will end here. If not, both players will withdraw and try to build up new units. As the production is ramped up and both players now have a good defence, the second peak at about 20min is broader.

There were about 20 games lasting longer than 25min, based on the uncommon situation that one player did not give up easily.

Gametype	*Realtime Strategy*
Uplink Rate	8kbit/s
Downlink Rate	8kbit/s per client
Session Time	see figure 7

Table 1: Variables for SC Model

4.2. Unreal Tournament

In Section 3.2 we found that the time signal of the traffic shows user related parameters. It is an interesting fact that there are situations where the upload traffic abruptly rises, i.e. such situations occur when returning to the game after a frag. Situations where both up- and download traffic jump from low to high, are due to level changes. As the starting period in a game can be quite crucial to the result of the game (gathering special artefacts such as shield belt and power upgrades will be easy for the player who enters first) this fact should be taken into account too.

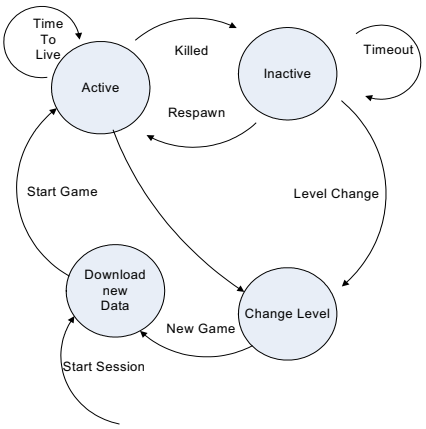

Figure 8: State diagram for a user model

The user model consists of four states. Starting from the beginning of a session via “New Game” or “Start Session” the user possibly has to download some level-data or program updates. After this accomplishment, the server will grant access and the client will start the game – the *Active* state is reached. Here the client will send at a rate corresponding to circle I in Figures 3b & 5. In getting killed the process changes to the *Inactive* state. Here the upload packet rate will drop to circle II in Figures 3b & 5 with the download parameters staying constant. Returning to the game, the client is now in the *Active* state again. At the end of each session, the model changes to the *Change Level* state. Here only very few data is sent and received, corresponding to circle III in Figures 3b & 5.

Next we analyzed the session length similar to SC. However as UT features different game styles, we have to treat them separately.

The simplest mode is *deathmatch*. Here every player tries to get the highest number of kills within a given time (sometimes there is also a frag limit). A *deathmatch* game normally has two end triggers, one is a maximum time counter and the other is a maximum frag number. If a player reaches this number of frags (i.e. 60), he automatically has won the game. Common time counter values for this game type are 15min and 30min.

The other two common modes which are also organised in a league, are *team-deathmatch* [12] and capture-the-flag. These two games are more tactical and normally last for 20min. The team which scores higher in this time span wins.

Recording a LAN party with many duels and 8h playtime, we extracted a PDF for the time between two deaths of a player (Fig. 9). We would explain the three peaks in the graph as follows: the peaks around 2-5sec correspond most likely to a spawn kill. If a player respawns, he only has a basic weapon. Therefore, the player is killed without resistance. The next peak we would explain as "normal" life time in a FPS game. 11-14sec is enough to gather some weapons and go roaming. The last peak is caused by a map control (one player is controlling the game for some period). One player can get hold of some goodies like the shield belt, extended armour and so on, extending his life time. We also tried to find some correlations in the sequence in which these three values occur, but our test sample was not large enough for useful information (ca. 350 kills).

The time a player needs to get back into the game is 0.9 - 1.1sec. These values were quite constant for one desktop, obviously depending on different hardware settings. The time needed for changing a level varied somewhat more. For small to medium levels it took 2.4 – 2.8sec for the client to load and enter a new level.

Gametype	*First Person Shooter*
Uplink Rate	varying - see fig 3, 5
Downlink Rate	varying - see fig 4, 5
Session Time	15, 20, 25 min
Time to Live	see fig. 9
Timeout	0.9 – 1.1 sec (server dependent)
Time for *Level Change*	2.4 – 3.8 sec (client dependent)

Table 2: Variables for UT User Model

5. M(ean) O(pinion) S(core)

In the following we describe the basic service behaviour to judge the maximum rates for a channel in terms of delay and packet loss

5.1. Star-Craft

In Star-Craft any lag in the network will be executed on every client, to keep the game synchronised. This means that if a client has a lag of 1sec, the other clients are stopped for this time period. Therefore, high lags are quite annoying but will not affect the game run too strongly. Up to 300ms delay, the game will stay playable well (Fig. 11). Adding packet loss, SC will not be so forgiving. As it is crucial for the game to stay synchronised, a packet loss will result in quite high traffic spikes up to link termination. At our test setup, 2% packet loss of one client killed most of the games.

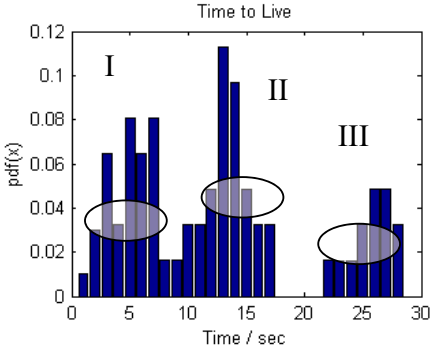

Figure 9: "Death rate" of players.

SC is a quite old game, in which limited bandwidth is not the problem. The game remained playable with about 32kbit/s (Fig. 10).

We found that delay normally does not change the outcome of the played tourney – also see [8] for WC3. But at about 300ms it became hard to manage big groups of units. This defines an upper limit for a service definition (Table 1).

5.2 Unreal Tournament 99

While Star-Craft represents a low consuming class of online games, UT 99 is more demanding with respect to network requirements. The logical connection between server and client is different to SC. In FPS there is a server defined which manages all important events such as the player's position-change and frags. This is reflected in the problem that a player with packet loss will appear jumping through the level on a different client.

A short lag in the network connection normally results in being killed. To make the games more playable there are prediction mechanisms at the client and at the server side. Normally the client has to report a user input to the server and waits for confirmation. With this method even a small delay would be visible. To make this delay invisible, the client assumes that the server accepted the input and executes the input immediately. While hiding delays, this results in asynchronous clients. This fact should always be kept in mind when judging MOS curves [6].

A UT player has to react fast and accurately. A fast reaction needs a low delay, an accurate movement nearly no jitter or packet loss. The following figures show that packet loss has to be seen as a function of the delay. At higher delays some small packet loss is not seen by the players. On the other hand, it is obvious that the "brain–hand" interface can handle delay much better than jitter and packet loss, which is easy to explain. If you have a

constant high delay, you can predict where the player will move and try to shoot him there. Adding jitter or packet loss will cause the player to change randomly his place (remember the server executes the input and sends the results) and even being killed without knowing why (server placed him into a bad place without notice).

The MOS picture (Fig. 12) shows that the service is quite sensitive to packet loss. This is on the one hand a result of the server's interacting with the client as we saw before and on the other hand a problem of our packet loss implementation. If the player engages an opponent, the traffic will rise but in the same way the equal distribution of the packet loss in the dummynet module will result in higher net loss of packets. Thus, in times when the player needs the best network conditions, he receives poor parameters.

Like Star-Craft also UT99 is quite an old game and therefore can be played with 32kbit/s (Fig. 10).

The impact of the delay on both the MOS and the frag-rate is unexpected. Literature [6] mentions that the frag-rate will lower if the delay rises. After a closer examination we found that the pro-gamer had an impact on our statistic. We saw that the pro-gamer could handle the increasing delay much better than the other players. Therefore, we conclude that it is not always a correct assumption to correlate higher delays with a lower frag-rate. A visual inspection of the game style showed that a normal player sticks to a small set of weapons while a good player will adopt his weapon choice to the network conditions (i.e. a rocket launcher, doing splash damage, will not suffer from lag). As our test sample only included one pro-gamer this can only be an assumption, but as these people will be the target group for marketing, this assumption has to be examined further. Figure 13 shows the "frag-share" for a five player session with different delay settings. The pro-gamer (E) pulls ahead when

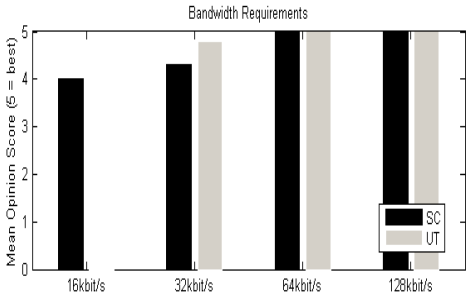

Figure 10: Bandwidth requirements for SC and UT

moving to higher delay. C's fondness for low precision weapons is seen as a peak around 120ms. The weakest player (A) is suffering most.

6. CONCLUSIONS

In the first part of this paper we introduced basics about RTS and FPS games. The two games chosen for experiments were *Unreal Tournament 99* and *Star-Craft*. It was shown that the behaviour on the packet level is quite similar to the games already studied in literature. Our interest was to adapt online-gaming-models to wireless needs. We extended these models with respect to the user-level, by analysing the session times. Combined with the service-parameters this allows cost estimation for wireless services. For RTS purely the session times seem to be sufficient for modelling, because the up- and down-rate model is constant. FPS game model requires some additional input parameters. Therefore, we propose a state

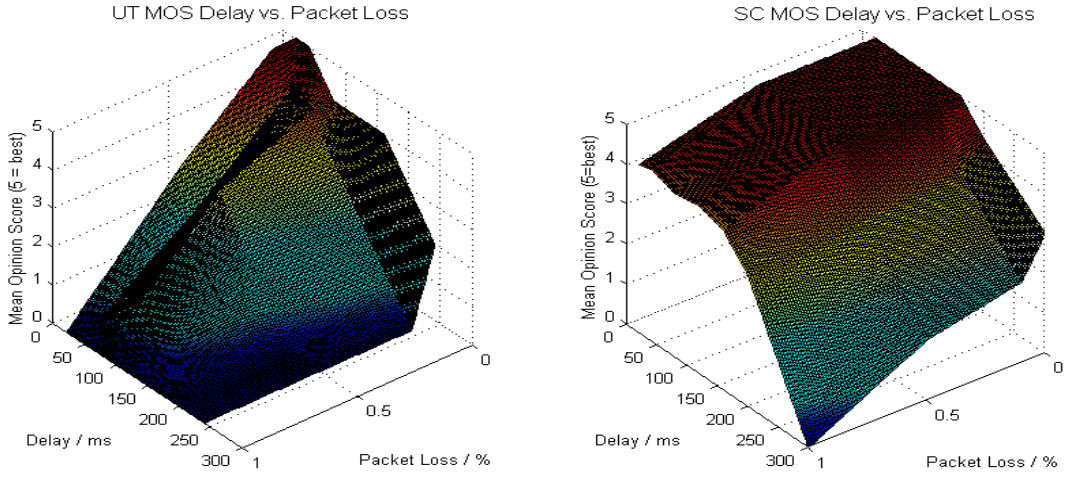

Figure 11,12: 3D MOS, delay vs. packet loss; Legend: 5-1 = good – worse; 0 unplayable

Game	*Unreal Tournament 99*	*Star Craft / Broodwar*
Service Type	Interactive Class	Conversational Class
Bandwidth	> 32kbit/s	> 32kbit/s
Delay	< 300ms	< 150ms
Packet Loss	< 1%	< 1.5%

Table 3: Possible Service Definition for UT and SC

model with the four states: *Active, Inactive, Level-Change* and *File-Download*. The model is styled for wireless networks, where resources are allocated dynamically. There abrupt rises of traffic load (ie changing from *Inactive* to *Active*) will cause service degradations.

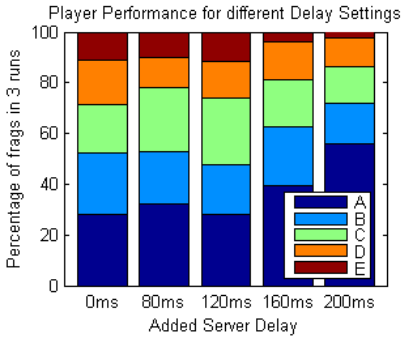

Figure 13: Player performance on different added delay

Because wireless networks normally suffer from both: delay and packet loss - the MOS figures were extended to 3D-plots over delay and packet loss. We deduced that the MOS is linked to the level of the players: a pro-gamer was able to handle much more lag.

Figures 14 and 15 show first measurements in an UMTS environment. Figure 14 shows the inter packet time of a SC UDP stream. It is shown that the UMTS network can easily handle constant sized packets. There is nearly no difference to the LAN environment values. The UMTS UT inter packet times differ strongly from the LAN values – It could be that the strongly varying packet size and rate is hard to handle for UMTS systems (could be a matter of traffic shaping in the GGSN or retransmissions at the RLC layer). The high probability that two packets have nearly 0ms inter arrival time assumes that there is some queuing going on somewhere.

But we can definitely conclude that UMTS is ready for real-time strategy games, FPS will need some tweaks.

7. FURTHER WORK

The next steps are to perform some measurements in UMTS environments and try to map those on adequate platforms. NIST.net is expected to work as a good emulation tool for this. Another option is the combination of the network demands (cost!) with the 3D MOS curves to find optimum regions for different networks, or even defining a service class named *mobile-gaming* for UMTS. SC got a MOS rating of 4 – at UMTS test runs.

Acknowledgments

Thanks to all the players who spent many hours on several weekends testing. Special thanks to Z0MBIE{DUST}.at and MadOne for some extended sessions.

Additional thanks go out to Olivia Nemethova for her input on MOS measurements.

REFERENCES

[1] Johannes Färber, "Network game traffic modelling," Proceedings of the first ACM workshop on Network and system support for games, April 2002

[2] http://www.ezunix.org/, „How to setup dummynet in FreeBSD“

[3] Tanja Lang, Grenville Armitage; “A Ns2 model for the Xbox System Link game Halo”

[3] Y. W. Bernier; “Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization.”; Game Developers Conference; Feb. 2001.

[4] T. Beigbeder, R. Coughlan, C. Lusher, and J. Plunkett; “The Effects of Packet Loss and Latency on Player Performance in Unreal Tournament 2003”; Major Qualifying Project MQP-MLC-NG03, WPI, May 2004.

[5] M. S. Borella; "Source models of network game traffic"; Proceedings of networld+interop '99, Las Vegas, NV, May 1999

[6] Grenville J. Armitage; “An Experimental Estimation of Latency Sensitivity In Multiplayer Quake 3”, Centre for Advanced Internet Architectures; Technical Report 030405A

[7] Tanja Lang, Grenville Armitage, Phillip Branch, Hwan-Yi Choo; ”A Synthetic Traffic Model for Half-Life”; Centre for Advanced Internet Architectures Swinburne University of Technology

[8] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu.; “The Effect of Latency on User Performance in Warcraft III”; ACM NetGames, May 2003.

[9] http://www.idsoftware.com/

[10] http://www.unreal-tournament.com/

[11] http://www.blizzard.com/sc

[12] http://www.wikipedia.org/

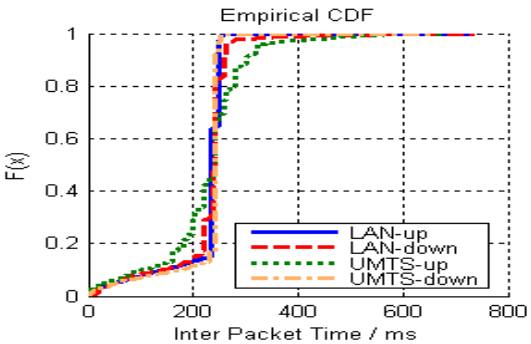

Figure 14: SC-Inter Packet Time UMTS/LAN

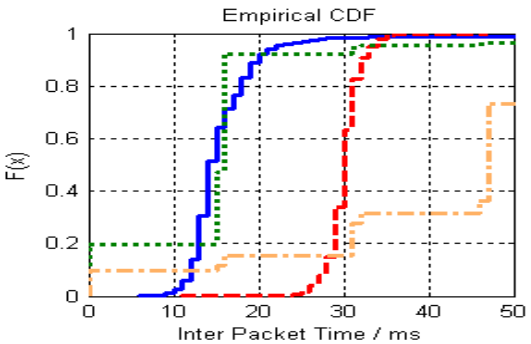

Figure 15: UT-Inter Packet Time UMTS/LAN