Ping Estimation in Enemy Territory

Alex Shoolman¹ Centre for Advanced Internet Architectures. Technical Report 070122A Swinburne University of Technology Melbourne, Australia alexshoolman@gmail.com

Abstract- This report analyses the round trip time estimation of the multiplayer game Return to Castle Wolfenstein: Enemy Territory. It describes the effect of various changes in server side snapshot rates, server side tick rates, as well as different round trip times added on the clients end. Enemy Territory (ET) is a very popular First Person Shooter (FPS) game even though it is quite old. The data that was used was drawn from client computers and an ET Server setup at The Centre for Advanced Internet Architectures (CAIA). Although this server was private, the game play was real with single and multiple players. We found that round trip times are often over-estimated in ET by as much as 50ms, as well as that the server side tick rates impact heavily on the estimated delay times given to its clients.

Keywords- Enemy Territory, snapshot rates, tick rates, CAIA

I. INTRODUCTION

In most first person shooter (FPS) games such as Enemy Territory (ET) [1], Half Life 2 [2] or Quake III [3] you are able to establish how much delay is between you and the server you're playing on. This paper's goal is to determine, at least for ET, just how accurate this given value is in respect to the real network delay.

In most online FPS games the architecture involved is a single server computer that is host to a number of client computers. These clients all transfer their current game commands at regular intervals to the server. The server receives all the game commands and calculates the game state according to various rules built into the game such as gravity, other client's actions and so on. Once this game state is calculated it is then transferred back to the clients. They accept it and display it on the screen for the player to view. Then the whole process repeats itself. The "ping" value or Round Trip Time (RTT) is what gives the client an idea of how long it takes a packet to travel to the server and back. If you are very far away from the server or have a slow connection a ping time of, say 300ms might be observed. It would be safe to say that the game play at this rate would be quite jerky, lag sometimes and most likely un-enjoyable. On the other hand if your computer is connected directly to the server through a switch you can easily expect ping times of <1ms. This will guarantee the smoothest possible play for the computer your on.

The purpose of this report is to determine how

accurate this estimated ping value is in regards to the actual measured network delay. It will investigate what values are displayed when set, known network delays are in place. It will also show what happens to these values when certain in game variables are altered. To be sure that the delay between the server and client is constant and accurate, computers are set up to simulate network delay by using a FreeBSD [4] based module called dummynet [5]. Dummynet is able to selectively control the bandwidth, packet loss rate or delay value on any link.

This report firstly explains how to set up and configure the network used to gather the data. This includes steps on how to set up the bridge and firewall computer as well as how to implement specific delay times. It also gives some test results that back up the findings in other reports [6] that show how accurate the dummynet application is. The test bed used includes four computers and a switch. One computer acts as a bridge and also controls the delay limitations throughout the tests. Another computer is set up to be the ET Server whilst the other two computers are the client computers.

There are four important variables that are altered throughout sections three and five that can be confusing. Below they are listed and explained clearly.

- Kernel tick rate The actual tick rate that the kernel runs at. (default of 1000Hz)
- ET Server tick rate The tick rate that the ET game server runs at. (default of 50ms)
- ET Server snapshot rate The rate at which the ET game server sends out game state updates. (default of 50ms)
- Client side command rate The upper limit at which the client sends its game data to the server. (default of 33.3ms)

Section three shows the results for the effect that altering the kernel tick rate has on the ping value. It is changed from 100Hz to 1000Hz in order to see if it impacts on the estimated ping value in ET.

Section four shows results for the various induced time delays on the client's side. The set delay between the server and the client is changed to different values to simulate a client that is physically further away, or that has a slower connection to the server. With the

¹ The author was an undergraduate engineering student on a short-term internship at CAIA while writing this report

various delays in place the ping estimation value is monitored to see if the server gives better or worse estimations based on the different delay time.

Section five shows the results regarding to the altering of the server side snapshot rate as well as altering the client side command rate. The final section draws conclusions based on the previous three sections.

II. SETTING UP THE NETWORK AND GATHERING DATA

A. Setting up the bridge and firewall computer.

First the bridge and firewall computer were set up. FreeBSD 6.1 [7] was installed onto a computer that had two network cards. The bridge [8], ipfw [9] and dummynet modules were loaded into the kernel using the following commands.

- > kldload bridge
- > kldload ipfw
- > kldload dummynet

Modules are dynamically loaded software libraries that interface with the kernel in order to add certain abilities to the computer. The bridge module for instance adds the ability to act as a bridge between two networks. Next bridging was enabled so that the two networks could communicate. This was done by setting the sysctl values.

> sysctl net.link.ether.bridge_cfg="sis0,fxp0" > sysctl net.link.ether.bridge.enable=1

The "sis0, fxp0" represents the name given to the two interfaces, in this case a Sis 900 10/100Base Tx (sis0) onboard network card and an Intel 82550 Pro/100 (fxp0) network card. The "1" instructs the system to enable the bridging ability. These sysctl values are configuration options used within FreeBSD and are required for the bridging, ipfw and dummynet application to function properly. Once these two sysctl values were set ipfw was enabled by entering the following commands:

> sysctl net.link.ether.ipfw=1
> sysctl net link ather bridge infu-

> sysctl net.link.ether.bridge.ipfw=1

The first command enables ipfw whilst the second enables ipfw to work in conjunction with the bridge. This is so that it not only connects two networks together, but also can act as a bandwidth limiter or delay inducer on that connection. The dummynet application was then enabled with the following commands. These were to allow data and ping commands to go to and from the ET Server during the tests.

- > *ipfw add 102 pipe 1 ip from 136.186.229.252 to any*
- > ipfw add 103 permit icmp from any to any
- > ipfw pipe 1 config bw 0 delay 0 plr 0

This last command was altered throughout the tests to set specific delay values on the data going to and from the ET Server. If a delay of 60ms was required the command was re-entered as:

> ipfw pipe 1 config bw 0 delay 60 plr 0

B. Setting up the Enemy Territory Server.

The ET Linux Server edition of ET was installed so that the computer could act as the main ET Server for the testing. The file to do this was:

et-linux-2.56-2.x86.run [10]

Once downloaded, it was executed with the following command.

> ./et-linux-2.56-2.x86.run

Once uncompressed, the installation was finished using the user directory under the folder name /enemyterritory. To make sure that the Server contained the most up to date code the following update file was applied:

et-linux-2.60-update.x86.run [11]

Once downloaded this was executed with the following command.

> ./ et-linux-2.60-update.x86.run

After a system restart, Gmod 1.2 was installed so that the server's recordings could be more easily extracted for analysis. Gmod 1.2 is a program that extracts the estimated ping times produced by the ET Server and then generates ping histograms. [12]

C. Setting up the Enemy Territory Clients.

Windows XP was the operating system used on the client's computers. In order for them to play ET they had the following file installed:

WolfET.exe [13]

Once through the (easy to follow) installation guide the patch was applied to bring the clients up to the same version as the server as well as assure that the most up to date code is used. The file below was installed to do just this.

ET_Patch_2_60.exe [14]

Once both clients were set up with windows and ET they were connected to the rest of the network as shown in Figure 1.

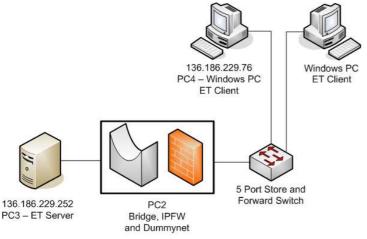


Figure 1: Network setup of server, clients and bridge/firewall.

This setup was then altered for the experiments in section five. When ET detects that the server and client are both on the same subnets, such as in the setup in Figure 1, it overrides the cl_maxpackets variable. In order to control this variable the set up was slightly altered so that the bridge became a router. Also in section five experiments only, one client was used instead of two. As shown in Figure 2, the server and client are on different networks and hence the ET Server does not change the cl_maxpackets variable automatically.

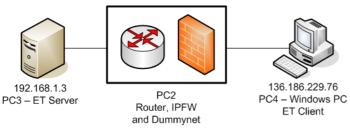


Figure 2: Network setup of server, clients and router/firewall.

D.Gathering Data.

The gathering of data was done on the ET Server computer. The Gmod 1.2 program was used to gather the ping times estimated by the ET server. The ping and tcpdump commands were also used to help analyse packet inter arrival times. This was to make sure the settings on the ET client's were correct. Tcpdump is a program that allows packet sniffing of the network. Large amounts of information can be gathered about general network activity, as well as specific computer-to-computer activity. The tcpdump and ping reports provided a contrast by letting us see what a standard ping and tcpdump returns as the network delay and what the ET server ping estimation returns. In all the tests, information from one client only was used from the generated log files. The reason two clients were used but only one client's data was analysed was to try and simulate real game experiences. In all tests the entire game was used as

data with bucket sizes of 1 and 2ms used.

Game lengths were very short, ranging from 5 minutes, 26 seconds to 18 minutes, 18 seconds. There were only ever up to two players connected, however they emulated real in game action as much as possible by completing tasks for that map, picking up items and killing one another. Other tests could be done with multiple clients connected from different locations and for over a longer period of time in order to increase the amount of data and hence its accuracy. This would also give it a much closer approximation to real life game play. However it was not our intention to do a full-scale test, this was simply an initial look into how ET estimates its client ping time based on limited game play

Initial tests were done to calibrate and determine the delay caused by having the bridge/firewall computer on the network. To do this, four tests were performed, each time with 100 pings at 1-second intervals:

Network Setup:	Median Ping Time
ET Server connected to ET Client via crossover cable.	0.2270ms
ET Server connected to ET Client through bridge. No ipfw enabled.	0.3265ms
ET Server connected to ET Client through bridge. Ipfw and dummynet enabled but with no pipe or delay values.	0.3085ms
ET Server connected to ET Client through bridge. Ipfw and dummynet enabled as well as a 500ms delay.	500.13ms

Table 1: Various delays recorded from the bridge/firewall computer.

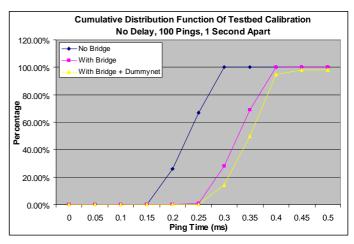


Figure 3:CDF of first three tests indicated in Table 1.

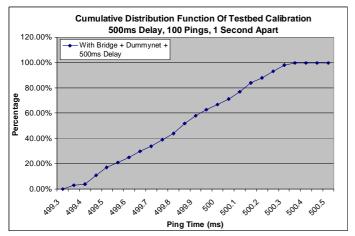


Figure 4:CDF of the last tests indicated in Table 1.

As Table 1, Figure 3 and Figure 4 show, the difference between having a bridge and firewall and not having one was minimal. Delays caused by having the bridge/firewall computer in between the ET Server and the ET Client are below the one millisecond mark. The main part of this one millisecond delay comes from the bridge computer having to receive and analyse the packet. It then has to apply the appropriate delay, packet loss or bandwidth restrictions before retransmitting. This process requires a number of CPU cycles for each packet that is analysed, and therefore increases the ping time between the two computers.

This error can be seen in the gradual build up of the round trip times in Table 1. As more and more layers are added, it takes more and more time for the computer to process each packet and thus adds to the overall delay. All four tests in Table 1 give a clear indication that measurements taken may be off by up to ± 1 ms. The ping of a client in a FPS game may vary a lot more than 1ms due to various demands that are placed on the computer, for example when large groups of clients are all in the one field of view. Due to this, if a delay is set using dummynet, for example 10ms, the actual delay is anywhere from 9ms to 11ms when you add the additional uncertainty of dummynet.

III. KERNEL TICK RATE

To simulate a "far away client", the bridge was configured to have a set delay of 73ms in one direction only. The ET Server computer was set to different kernel tick rates for each test; one at 100Hz and the other at 1000Hz. This was done by adding the following to the bottom line of the /boot/loader.conf file and then restarting the machine.

After restarting, the ET server was started with the following command. Two clients connected, both playing two times. The first was for 10 minutes at 100Hz tick rate and the other for 18 minutes at 1000Hz.

> ./etded +set dedicated 2 +set fs_game etpro +setsv_punkbuster 1 +set net_port 27960 +exec server.cfg +set fs_homepath 27960 +set fs_game etgmod

When starting up the ET Server with the above command, a few extra things where added to do such things as enable punkbuster – an add-on to stop clients from cheating – set the path of the log file and to enable the Gmod 1.2 program. The two separate tests where then analysed using the Gmodstat [15] program. This takes the outputted log files from the ET Server and converts them into a histogram (see Figure 5).

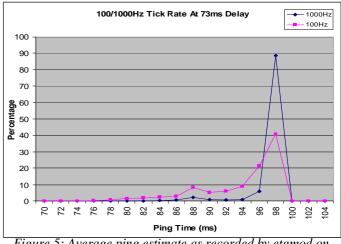


Figure 5: Average ping estimate as recorded by etgmod on the ET Server for different Server tick rates.

As shown in Figure 5, with the kernel set to 1000Hz the ET Server had a much more accurate estimate as opposed to the 100Hz setting. Although in both cases some readings occurred at the 88ms mark, most appeared to be at exactly 98ms. This represents an over estimation as the set delay could not have been any more than 73ms ± 1 ms.

IV. ARTIFICIALLY CREATED CLIENT SIDE DELAYS.

The initial delay in section III was set and verified (via tcpdump and ping commands at both ends) to be $73\text{ms} \pm 1\text{ms}$. The ET Server ping estimate however displayed major peaks at around 98ms. To check that this wasn't simply a specific delay setting that "broke" the estimation code, three more short games were completed at the 1000Hz setting but with varied delays. There was a 33ms, 76ms and 133ms delay programmed into dummynet and tested via ping and tcpdump. The aim was to find some common trend as to how the ET Server estimated the packet round trip time as a function of actual network induced delay time.

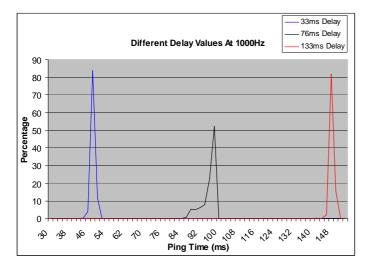


Figure 6: Average ping estimate as recorded by etgmod on the ET Server for different delay times.

The difference between the actual delay between the two computers and the estimated delay given to the client by the ET Server varies as shown in Figure 6. Both 33ms and 133ms were 15ms \pm 1ms off from their configured delay values. Whilst for the 76ms delay case, we saw a 22ms \pm 1ms difference. It is clear from this that the estimation given by the ET Server can be at least 22ms off what the actual network delay is. This could be investigated further by determining how exactly the code estimates the ping time and thus, where the break down occurs.

$V.SERVER/CLIENT\ SIDE\ SNAPSHOT/CMD\ RATES.$

The last variables to be tested were the server side snapshot rate and client side command rates. Both were changed separately whilst ping times were estimated by Gmod 1.2. The tests were conducted with only one client and were all completed over a short game time of roughly 10 minutes. The ping estimations were then analysed by Gmodstat.

In all the Snaps = 10 scenarios the artificial delay was turned off and packets were allowed to propagate at their maximum rate with only the bridge in between for monitoring purposes. In the Snaps = 20 cases the artificial delay was set to 60ms. If the delay is set to 0ms, Gmod 1.2 doesn't record any ping values as it records 0ms for too long and dismisses them as errors.

During the tests the network traffic was monitored with tcpdump, and the PPS rates were extracted using pkthisto [16]. Results shown in Figure 7 indicate that when set to a specific value, the snapshot rate is very consistent with that value. It is also worth mentioning that because of the ET Server tick rate, you may only set the snapshot rate to 10, or multiples of 10. If you set it to anything outside this it will be rounded down to best fit. For example, if we choose a snapshot rate of 19, you will see the same results as shown for snaps = 10.

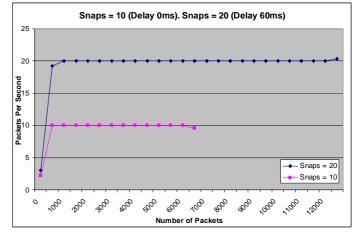


Figure 7: The packet per second readings for different server snapshot rates.

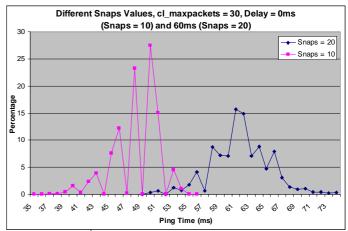


Figure 8: ET's ping estimation times as recorded by etgmod on the ET Server for different snapshot rates.

In Figure 8, it is clear that when the ET Server snapshot rate is set to 10 instead of its default 20, an approximate 50ms was added to the ping estimation. The jagged form of the Snaps = 10 distribution shows that ping times of specific values never occur. Although they both have peaks, the Snaps = 10 case has distinct values that report 0% pings at specific times.

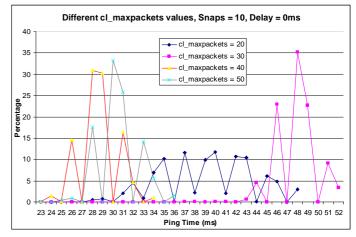


Figure 9: Average ping estimate as recorded by etgmod on the ET Server for different client side cmd rates.

CAIA Technical Report 070122A

The cl maxpackets variable is the limit (in packets per second) that the client sends command packets to the server. This is also called the client side command rate. In Figure 9 the server side snapshot rate was 10. The various values assigned to cl_maxpackets changes the ping estimation of the ET Server quite a lot. In addition to the cl_maxpackets variable there was also another variable called com_maxfps which set an upper limit on the number of frames per second (FPS) that the clients computer could generate. This variable, if set below the cl_maxpackets variable, would limit the packets per second rate from the client to the server. Due to this finding, the variable com_maxfps was set to its default value of 85 (frames per second) for all of the tests. As cl_maxpackets never exceeded this value there was no problem. There was no actual network delay during this test.

The next test was done at a snapshot rate of 20 and an artificial delay of 60ms was put in between because otherwise the Gmodstat program did not identify "active play" as the ping time was 0ms for far too often.

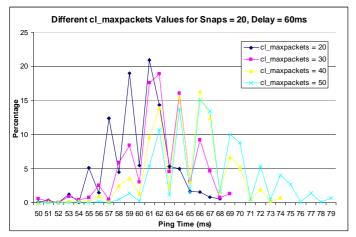


Figure 10: Average ping estimate as recorded by etgmod on the ET Server for different cl_maxpacket rates (snaps = 20)

As shown in Figure 10, the altering of $cl_maxpackets$ creates some deviance in the ping estimation. Based on these results and the ones with Snaps = 10, it is hard to say exactly what effect changing the client side command rate has on the ping estimation. In the Snaps = 20 case shown in Figure 10 we seem to see that as the cl_maxpackets value increases, the ping estimation gets further and further off its actual value. However in the Snaps = 10 scenario as the cl_maxpackets values go up the ping estimations sometimes get closer to their true values.

To make sure that the values being set with the cl_maxpackets did actually correspond to the clients PPS rate a graph was plotted similar to Figure 7, mapping out the recorded PPS rates for each test done by tcpdump. Shown in Figures 11 and 12, it is quite clear that the client side PPS rate isn't as precise as the server side snapshot rate graph in Figure 7.

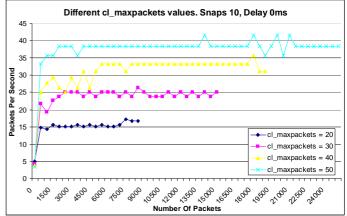


Figure 11: Different PPS rates for each client side cl_maxpackets value. Snaps = 10, Delay = 0ms.

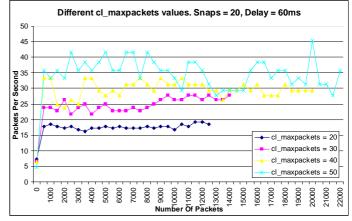


Figure 12: Different PPS rates for each client side cl_maxpackets value. Snaps = 20, Delay = 60ms.

Over the course of both of the tests shown above in Figures 11 and 12, it was clear that although reasonably steady, the PPS rates do not match the set cl_maxpackets rate. The actual measured PPS rate was off by 2-12 PPS from the desired rate. These tests were also conducted, as indicated by Figure 2, in an isolated network with the server and client being on completely different networks. This was to avoid the ET Server from overriding the cl_maxpackets value we set due to the server and client being on the same network.

It is clear that in each different test, as we increased the cl_maxpackets value the PPS rate increased as well. Although the PPS values don't match the values we set, we can still see what happens to the estimated ping time (Figures 9 and 10) as the cl_maxpackets rate increases. With a server side snapshot rate of 10 we can see that the ping estimation in Figure 9 jumps all over the place. It is not at all like Figure 10 where as the cl_maxpackets rate increases steadily, so does the ping estimation time. Figure 9 seems to follow this trend until cl_maxpackets hits 40 where upon this, it changes the ping estimation to something lower than before.

VI. CONCLUSION

This paper gives a detailed analysis of the ping estimation generated by an Enemy Territory Server when different variables are changed. Data was gathered within CAIA using networked computers to simulate real world scenarios. One computer was altered to perform as a bridge and produce artificial delays. The estimation code that is used to give round trip times back to the clients was found to be different to the RTT that was set on the bridge computer. These differences were based on the actual network path delay. Other set values were also found to be added to the round trip times when certain variables were changed such as the server side snapshot rate.

Also shown was the effect that the tick rate of the computer running the ET Server has on the estimated RTT. As the tick rate in hertz increases, the server is more able to single in on the RTT. However, this result is often still incorrect due to errors in the actual estimation code.

Additional tests were done to analyse the effects of what happens when you changed the client side command update rate or the server side snapshot rate. These indicated that changing the server side snapshot rate to 10 instead of its default 20, caused the server's RTT measurement to be overestimated by about 50ms. Increasing the client side command rate gave the impression that the ping estimation was also increased as well. However there is also evidence to discount this in the case of Figure 9. Changing the client side command rate does alter the ping estimation, however in exactly what way cannot be stated without performing more detailed tests.

An investigation into how exactly the ET Server generates its RTT estimates would be needed to identify the reasons as to why it adds additional values to its estimation of the RTT. An increased number of clients that play for more time and are actually large physical distances away from the server would also add to making the analysed data more accurate to real online game play.

ACKNOWLEDGMENTS

Thanks go to the CAIA members who gave up their time to participate in playing Enemy Territory.

References

- [1] "Return to Castle Wolfenstein: Enemy Territory", activision main website, <u>http://games.activision.com/games/wolfenstein/</u>.
- [2] "Half-Life 2", Valve main website, http://www.valvesoftware.com/projects.htm.
- [3] "Quake 3", idSoftware main website, http://www.idsoftware.com/.
- [4] FreeBSD 6.1, main website, <u>http://www.freebsd.org/</u>.
- [5] Dummynet,
 - http://www.freebsd.org/cgi/man.cgi?query=dummynet&sektion=4
- [6] Vanhonacker,W. A., "Evaluation of the FreeBSD dummynet network performance simulation tool on a Pentium 4-based Ethernet Bridge", (<u>http://caia.swin.edu.au/reports/031202A/CAIA-TR-031202A.pdf</u>).
- [7] More information pertaining to the installation of FreeBSD can be found at this web site, Chapter 2 Installing FreeBSD of, the FreeBSD Handbook: <u>http://www.freebsd.org/doc/en_US.ISO8859-</u> <u>1/books/handbook/install.html</u>.
- [8] More information on bridge and its abilities can be found at this web site, FreeBSD Man Pages, Bridging Support: <u>http://www.freebsd.org/cgi/man.cgi?query=bridge&apropos=0&sekti</u> <u>on=0&manpath=FreeBSD+6.1-RELEASE&format=html</u> (as of Sept 2006).
- [9] More information on ipfw and its abilities can be found at this web site, FreeBSD Man Pages, IP firewall and traffic shaper control program <u>http://www.freebsd.org/cgi/man.cgi?query=ipfw&apropos=0&sektion</u> <u>=0&manpath=FreeBSD+6.1-RELEASE&format=html</u> (as of Sept 2006).
- [10] This install file for the Linux version of ET can be found here, FilePlanet <u>http://www.fileplanet.com/124801/120000/fileinfo/Return-to-Castle-Wolfenstein:-Enemy-Territory-Client-v2.60-%5BLinux%5D</u> (as of Sept 2006).
- [11] This update file may be downloaded from here, FilePlanet, Wolfenstein: Enemy Territory v2.60 Patch: <u>http://www.fileplanet.com/126607/120000/fileinfo/Wolfenstein:-</u> <u>Enemy-Territory-v2.60-Patch-%5BLinux%5D</u> (as of Sept 2006).
- [12] More information and instructions on how to install the Gmod 1.2 program can be found here, The CAIA Website, Gmod Overview: <u>http://caia.swin.edu.au/genius/tools/gmod/</u> (as of Jan 2007)
- [13] This install file for the client, Windows version of ET can be found here, FilePlanet, Wolfenstein: Enemy Territory Client v2.6: <u>http://www.fileplanet.com/124800/120000/fileinfo/Return-to-Castle-Wolfenstein:-Enemy-Territory-Client-v2.6</u> (as of Sept 2006).
- [14] This patch file can be downloaded from this website, FilePlanet, Wolfenstein: Enemy Territory 2.60 Patch: <u>http://www.fileplanet.com/130185/130000/fileinfo/Wolfenstein:-Enemy-Territory-2.60-Patch</u> (as of Sept 2006).
- [15] Gmodstat download: <u>http://caia.swin.edu.au/genius/tools/gmodstat-0.2.1.tar.gz</u> (as of August 2006).
- [16] Information on pkthisto available here, CAIA Tools, pkthisto main page: <u>http://caia.swin.edu.au/genius/tools/pkthisto/</u> (as of January 2007)