Netspeed Tutorial (UT)

From UnrealAdminWiki

Originally written by TNSe

Contents

- 1 Introduction
- 2 What is Tickrate?
- 3 Ping
- 4 Netspeed
- 5 OMG, I gotta set netspeed to 20000 immediately
  - 5.1 Server Admin Notice
- 6 So, lets put this together
- 7 Geez, we must have tickrate 100
  - 7.1 Server Admin Notice
- 8 Does tickrate affect weapons?
- 9 Ok, What Now ?
- 10 Addendum by El_Muerte
- 11 Addendum by Azura
- 12 Addendum by nogginBasher

Introduction

Ok, I just wanted to write some boring facts about the boring thing that bores us all.

Tickrate & Netspeed.

Note: this document only applies to Unreal Tournament, for other games it might have different results.

What is Tickrate?

What effect does it have in UT? Many ppl have seriously interesting views on this that easily can be put in parallel with the quotes from certain presidents in USA.

First of all. **Higher tickrate does not improve ping**. Very common misconception. What it **actually** does, is to reduce the time between each time the server handles incoming data, and sends out new data. So let me explain it in more complex terms.

By first explaining what tickrate actually is, I hope to have made you so confused you will believe the rest of the things I write in this document.

Tickrate is actually '*very very complex!* It is the "Frames Per Second" the server is running at. Yes, it is so complex. A tickrate of 20 means that the server is doing 20 frames of action per second. To make it even more complex, lets just say, that it processes data 20 times a second. (Player movement, player firing, player deaths, player bitching, player lagging, and the other more unimportant things 20 times a second). So. 1 Tick = 1 Frame. The server tickrate is the "framerate" at which the server is running on.

Ok, now that you are confused by the tickrate term, lets see how far it is between each tick update. since it says pr second, we will use 1 second, which is 1000 milliseconds. it's Per. so we gotta divide somewhere, let's try the following. 1000ms/20 = 50ms. OK! it might seem like there is 50 milliseconds between each tick. Let's try it the other way. 50ms * 20 = 1000ms. Oh yes, we reversed the equation and got what we started with. That means. A tickrate of 20, gives 50ms between each update on server.

Ping

Lets have a ping of 60 in DOS to a certain server. This means a packet of data uses 60 milliseconds to go from your superfantastic computer to the crappy tickrate 20 server in Uganda and back. In other words. Now because of the complexity of the net, this doesn't mean it uses 30ms to reach the server, and 30ms to travel back. (Asymmetric routing) Someone else will have to explain why this is so. But lets assume that this is actually the true thing this time. The data DOES use 30ms to travel from your computer to the server, and 30ms to reach your computer again. Let's put in some more numbers. The tickrate! We know that it's 50ms between each tick on server. So let's take worst case. The packet of data we send, uses 30ms to reach server. The server has just completed a tick before the data arrived, so the data has to wait for 50ms. Ok, data has waited 50ms and been processed, and now leaves the server for the client to happily enjoy. This takes another 30ms. So to add it up, the packet used 30+50+30 = 110ms to travel from client to server, be processed on server, and return to client.

Another thing that affects result is your own framerate. Your computer will not handle data while it's busy telling the video card what to show you. A framerate of 100fps means that it's another 1000ms/100 = 10ms between each time it can handle data.

You thought that was all? You were wrong. Again. Of course

The DOS ping usually only sends a small packet, of like 32 bytes. If only UT sent that little The more data, the higher the ping gets. Go ahead and try it yourself.

```
PING [server] -l 32 Pings the server you choose with a 32byte packet.
```

Example:

```
C:>ping ut.online.no -l 32
Pinging ut.online.no [193.213.112.70] with 32 bytes of data:
```

```
Reply from 193.213.112.70: bytes=32 time=35ms TTL=116
Reply from 193.213.112.70: bytes=32 time=34ms TTL=116
Reply from 193.213.112.70: bytes=32 time=34ms TTL=116
Reply from 193.213.112.70: bytes=32 time=34ms TTL=116
```

This looks good! Now lets try with a larger packet, like 512 bytes.

```
PING [server] -l 512
```

Example:

```
C:>ping ut.online.no -l 512
Pinging ut.online.no [193.213.112.70] with 512 bytes of data:
```

```
Reply from 193.213.112.70: bytes=512 time=156ms TTL=116
Reply from 193.213.112.70: bytes=512 time=155ms TTL=116
Reply from 193.213.112.70: bytes=512 time=155ms TTL=116
Reply from 193.213.112.70: bytes=512 time=155ms TTL=116
```

Oops, it suddenly didn't look too good anymore right? But then again, I only have a fancy 64kbit isdn line. Mileage may vary depending on your connection.

Netspeed

What is netspeed good for? What does it do? Is Bin Laden dead? Unfortunately, I can only answer the 2 first questions. Well, seriously, the 2 first questions is actually 1 question... Err, anyway.

The netspeed decides how much data you want to send to the server each second. A netspeed of 5000 will try to send 5000 bytes of data each second. And yes, for the smart ones out there that already guessed it, netspeed of 13690 will send 13690 bytes pr second. Also, it tells the server how much data you want to receive each second. The servers seem to not allow going under 2000, clients have a minimum of 500.

Some interesting aspect with the netspeed, is that it limits your framerate. Your maximum expected framerate online with netspeed 5000 is 5000/64 = 78fps. You should not get more, you will often get less. So why /64 you ask? Well, it's kinda simple. Each time your computer does an update, it sends about 64 bytes of data. So the great doods at epic thought, let's do netspeed/64 and limit framerate that way, so the client does not exceed netspeed bytes sent pr second.

For those of you thinking:

OMG, I gotta set netspeed to 20000 immediately

let me just inform you that your hardware probably can't hold an even 20000/64 = 312fps average during an entire match anyway. Another bottleneck here, is your internet connection. If you got a state-of-the-art 9600 modem, it can only send and receive 9600/8 = 1200 bytes/second anyway, and what happens if you exceed that limit, is that data must wait before it can be transmitted to/from you. This causes the wellknown "ping spike". So use netspeed to make sure your connection does not get overloaded and "spiked". If the spikes last too long, you might even get packetloss, since the packets get pissed off waiting in queues and kill themselves.

I found a nice trick myself. I put my netspeed to refreshrate*64 (and then + some for a nicer rounder number). I got 85hz refresh, *64 = 5440, so I set netspeed to 5500. This gives a nice flickerfree experience when I play online. To be bloody honest, I had been experimenting with netspeeds for a while until I found 5500 to be most pleasant. I didn't know about this *64 thing at the time. Weird coincidence or facts, you try yourself.

Now I've said all about netspeed and effects on client to server. Let's take it the other way. What happens if the server wants to tell me too much. It can only tell me 5500 bytes pr second. The solution is quite simple. The server stops telling you stuff it feels is nonimportant. This has been carefully hac... errr balanced by the brainmonsters at Epic to give a great online gaming experience. Most important stuff like other players & bots have priority to be sent first. Then comes less important things, like projectiles & effects. So if the server has too much to tell you, it will put it off until it sees you have enough bandwidth to receive it. Thus, you get the infamous "rockets out of thin air"-syndrome. If it has to wait sending stuff too long, you might not ever receive it. The result of this is you going dead from invisible stuff, this is often referred to as "WTF??!" by the players affected.

Server Admin Notice

You got an option in your .ini file under `[IpDrv.TcpNetDriver]` called `MaxClientRate`. The default value of this is usually 20000, which means, the client can maximum get 20000 bytes *FROM* server per second. By reducing this value, you can actually optimize the server bandwidth. If you are planning on running a 10 player server on a 512/512 kbit sdsl line, you will have 64kbyte up/down, so setting MaxClientRate to 64000/10 = 6400 means you will reduce the chance of having packetloss. This setting will also limit the client upload rate, forcing him to keep a netspeed between 2000 and MaxClientRate.

So, lets put this together

As we all know, servers with *high* tickrate is *the* ownage thing out there. It's the best stuff that happened since man invented the transistor. Or is it?

Lets say we got my reasonable netspeed of 5500. Each tick on a tickrate 20 server, I'm allowed to receive 5500/20 = 275 bytes of data. That is rarely a problem unless there is too many players or too much spam going on.

Ok, now lets try increasing the tickrate to 40. I'm suddenly only allowed to receive 5500/40 = 137.5 bytes of data pr tick. This could easily start to create trouble for me, as the server starts deciding things aren't important enough to send to me.

I'll tell you about the goodies of higher tickrates as well. The first thing is that the server responds more often (Tickrate 20, 50ms between each frame, Tickrate 40, 1000/40 = 25ms between each frame). And the second thing is that other players (and yourself of course, DUH!) are updated more often. with tickrate 40 vs 20, everything is actually updated twice as often!

You've probably been missing some images in this document. Worry no more, here are some super images!

A small debriefing of what these pictures mean. Each image of my model is 1 tick on the server. So that means, each time you see my model, that is where the server said I was.


Tickrate 5


Tickrate 15


Tickrate 20


Tickrate 30


Tickrate 40


Tickrate 50


Tickrate 100

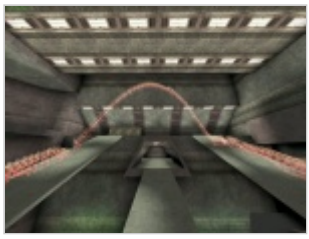
So what does this mean? Hint: You can only hit people where the server say they are. Look at tickrate 5, and see, to hit me, you almost gotta be lucky. At Tickrate 100, you gotta be bad to avoid hitting me. Some more pics.

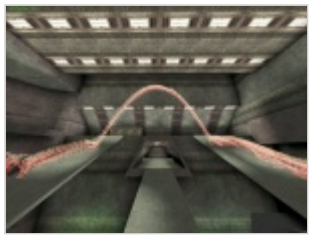

Tickrate 5


Tickrate 20


Tickrate 30


Tickrate 40

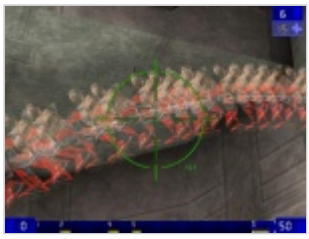

Tickrate 50


Tickrate 100

To say it again YOU HAVE TO HIT ONE OF THE "SHADOWS" OF MY MODEL FOR THE SERVER TO REGISTER A HIT.

And yes, hitting someone midair on tickrate 20 is harder than with tickrate 50. Definately. And 2 more


Tickrate 20


Tickrate 40

So now all of you are saying:

Geez, we must have tickrate 100

Nah, if you can't hit with tickrate 20, you have no skillz. High Tickrate = no skillz. Also remember what I said about data pr second. Higher tickrate = more data sent from server. You could start getting invisible rockets.

Now over to something really interesting. Let me show you the pictures first this time.


Tickrate 30, Netspeed 5500


Tickrate 30, Netspeed 1000

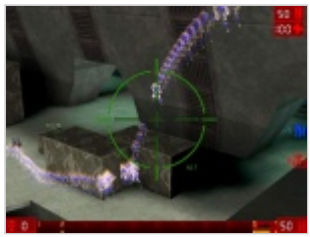

Tickrate 30, Netspeed 500

And it's still the same, each "shadow" of my model = where it is possible to hit me. And yes. messing with your netspeed WILL affect how "easy" you are to hit on server. Notice how the server updates your animation several times while you still are in the same place. The reason for this, take example with 500 netspeed, is that you are only sending 500/64 ~= 8 updates pr second to the server, while the server is doing 30 updates pr second, which means it will put you in the same position ~4 times in a row! Meanwhile, all other clients will see you fall in a nice curve, and shoot straight through you.

Here comes some pictures from a case where a player has intermittant outgoing packetloss (Movement data does not reach the server.) This could cause several problems, all ranging from total loss of data, to data coming in wrong order. All pictures were taken with Tickrate 30 on the server.

Tickrate 30, Packetloss, Pic 1

Tickrate 30, Packetloss, Pic 2



Tickrate 30, Packetloss, Pic 3

Tickrate 30, Packetloss, Pic 4

Pic 1 shows clearly what happens when the client is having a pure outgoing packetloss, simply, packets do not reach server, and are stopped somewhere Pic 2 is interesting, since it shows that the player with packetloss is CLEARLY incorrectly offsetted compared to what MY client is predicting. As you see, the player with packetloss is actually a lot higher up than my client is showing. The cause for this could be that the packetloss player got packetloss just as he jumped, this means the server didn't register any jump before he had started, and therefore told me too late that he did jump. so the entire flight path has wrong timing. Pic 3 is another example of what happens. Big holes in the movements. Pic 4 as well.

So what can be done with this? Nothing. This is due to the way UT behaves, it allows the client to control its own movement (to a certain degree) without server intervention. This is done to allow more smoother gameplay.If the client is too much out of order compared to what server means he should be, he will be moved back.

- Some people were missing an explanation for F1/F6 (stat net) pings.

Here is a rude formula to show the relation of DOS Ping, F1 Ping and F6 ping. It is not overly accurate, but you will hopefully see that there is a relation between them.

```
f1 ping = dos ping + (1000/fps + 0,5*1000/tickrate)*0,5
f6 ping = dos ping + 1000/tickrate + 0,5*1000/fps
```

Tickrate is the tickrate of the server, and fps is the framerate of your client. Remember that the framerate on clients can be limited by using netspeed. Netspeed 5500 gives ~85fps max.

For those of you who don't want a more indepth explanation of these formulas, skip over the following part.

The dos ping is used as a base for how long a packet needs to travel forth & back. This is the minimum latency needed to complete a ping-pong. First I will explain F6, since it is simpler. F6 is the response time for Client->Server->Client. Client requests it from Server, at server it has to wait a tick before it is sent back to client (= 1000/tickrate). Then it is sent back to client, and client cannot use this reply until (in average) half the time it takes to render a frame. (0.5*1000/fps). F1 is Server->Client->Server response time. Server asks client for a ping reply, Client waits 1 frame to answer (1000/fps), and server needs in average half a tick to process the reply. (0,5*1000/tickrate). And for some reason, Digital Extremes/Epic Games divided the time this takes in 2. (*0,5). (This can be found in the uscript code). Hope this confuses you who should have skipped this section.

Server Admin Notice

Increasing or Lowering the tickrate has documented effect on the gameplay (heck, what do you think this document is about?). Also, I'd like to point out the following. Higher tickrate = higher CPU usage! So setting your blazing P-2 350MHZ 32player max superserver to tickrate 100 is rather a bad idea. With higher tickrates, the bandwidth requirements go up as well. I'd recommend keeping tickrates at their default if you want more than 10 players on the server... A final note to those running Linux servers. For some reason, the guys at Loki mindlessly translated the UT code from Win32 based to Linux based without really thinking over the sideeffects it had. Linux is not as good as Windows at something which I can't explain properly without having 2million ppl rubbing penguins on my window telling me how I'm a microsoft lover. Back to the point. Linux isn't so exact at getting the tickrate correctly. This means a tickrate of 20 on a linux server might actually be more like 15... Which is why I'd nearly ask you linux phreaks to either get Win32 based server, or ... sigh, up the tickrate slightly. Just for a final note... WinNT4 SP6 based servers are the ones I like best. They provide stable gameplay. Win2k based servers give occational pingspikes, and Linux based servers can be a bit dodgy (stuff go through people). Never tried Mac Based servers. and Win98 based servers probably gotta be rebooted every day or so, most likely not a good alternative

Important: the above statement about Linux servers is not completely true, read the Addendum for more information.

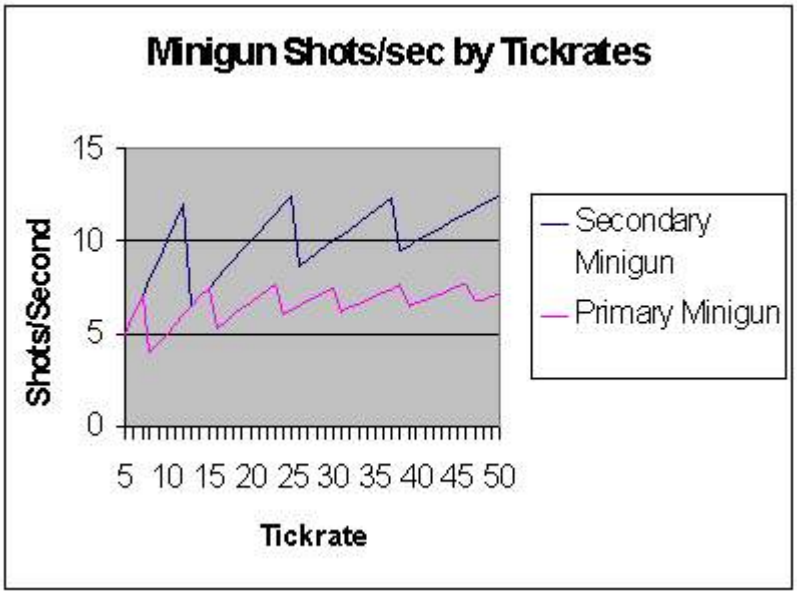
Does tickrate affect weapons?

Good question. Answer is. Yes. The affected weapons are:

- Enforcer
- Sniper Rifle
- Shock Rifle Primary
- Minigun
- Pulse Gun Secondary

Enforcer, Sniper & Shock are only affected in the way that hitboxes are more regularly updated (see pictures above). Also your aiming angle is used more often on the server, causing the entire thing to be just more accurate.

Now, the minigun is a nasty thing when it comes to tickrate. I'll simply put up a little pic here:



So what does this mean? Actually it's quite simple, read the stuff on the diagram. The minigun fires more or less bullets depending on the tickrate of the server. A tickrate of 20 will give you ~10 bullets/second, while increasing to 25 could theoretically give you 12.5 bullets/second. a 25% increase in damage.35 tickrate gives about 11 bullets/second. Remember though, this is entirely theoretical. The tickrate is not perfect on the server, it will vary a bit. But it's interesting enough as it shows on the picture. Different tickrates can be used to manipulate the power of the minigun. This is one of the reasons I think tickrate should stay at a default of 20. Simply to avoid minigun being overpowered.

And what about the pulse gun? Aside from the more accurate hit control, it grows faster. Yup. You heard me. The pulse beam extends with 1 segment (it consists of 9) per tick. This means it will use 50ms*8 = 400ms to reach full length at tickrate 20. At tickrate 40, it will use 25ms*8 = 200ms to reach full length. So if you are playing on a higher tickrate, the pulse secondary goes from being a close combat weapon, to becoming a very effective medium range weapon. Another thing is that the Plasma accumulates damage potential as long as it isn't in contact with any player. With higher tickrates, it can more often change state from being "in contact" and "not in contact", creating more damage as result.

So I hope this can explain why minigun & pulse are more effective on lans... It's not only the lower ping & more often hit registration of the weapons, but also side effects from the higher default tickrate of 35. I would suggest servers keeping the tickrate of 20 to avoid messing up the weapon balance.

Ok, What Now ?

A ton of thanks to Mongo for telling me a bunch of stuff I really didn't have to know. Because of all this stuff he told me, I had to write this document just to save space in my head for more important things. I also hope he has learnt a lot himself from working with UT's networking code, atleast enough to avoid some of the worst pitfalls encountered with UT, in upcoming games

Also thanks to Garfield the NUBie (He hates being called NUB for an unknown reason) who always has to tell me not all germans are bratwurst. (Why do all germans believe that they aren't) Also we have discussed tons of things tons of times. Usually ending up at the same conclusion every time. Ping stinks.

Addendum by El_Muerte

About the Tickrate and Linux. This is not caused by the "mindlessly translation" to linux by Loki. In fact, the Loki guys did a mighty fine job porting UT to Linux. The Linux trickrate issue is caused by the behavior of Linux schedular. This is easy to fix if you are willing to rebuild your Linux kernel.

For 2.4 kernels you can download the variable HZ patch here. You may also want to check out the other Linux 2.4 kernel patches made by RML.

For 2.6 kernels you do not need to apply any patch, the above patch has been merged into the 2.4 kernel.

For older kernel, you really should need to upgrade to a newer kernel, you're missing out a lot.

-Edit: if this is true why do newer linux servers still not put out the tickrate you set them to?

Addendum by Azura

TNSe has made a mutator to fix the phenomenon where certain weapons are boosted by an increase in tickrate. It can be downloaded here :

CB Mutator for UT NW FallCup 2003 by TNSe (http://www.clanbase.com/miscdl.php?did=797)

Addendum by NogginBasher

UTPure's PureStats will also keep the weapons at their default strength.

Retrieved from "https://wiki.unrealadmin.org/Netspeed_Tutorial_%28UT%29"

Categories: UT

- This page was last modified 03:38, 11 April 2012.