# UploadBandwidthThrottler工作流程

一、各变量的含义

1、m\_StandardOrder\_list：标准列表，可发送控制包和标准包

2、

二、节流阀工作原理

1、统计一个循环经过的时间timeSinceLastLoop，再统计一个循环发送的字节数bytesToSpend，计算出传输速率datarate=bytesToSpend/timeSinceLastLoop，与允许传输速率比较，快了则休眠一段时间。需要注意的是，设定的速率是以秒为单位的，而循环里的计时则是以毫秒为单位的，字节数也是有以Byte为单位的，有以KB为单位的，需要换算。

假设限速300K/S，则换算成B/ms则为300\*1024/10000=307b/ms。如果前200ms传输了200K，则速率达到了1MB/S，还剩100K还需100ms，休眠时间则为1000-300=700毫秒。当然一个循环需要时间没有这么多，（调试跟踪到底需要多少时间）假设需要20ms，则能发送的字节为6K。如果上一个循环发送了7K字节，则本循环仅需要发送5K，需要时间为5/6\*20=17ms，则本次循环需要休眠的时间为3ms。而如果上一个循环发送了5K字节，则本循环需要发送7K，则本次循环不需要休眠，相反可以提高传输速率到350K/S。

假设限速300K/S，上一个循环到现在经历了20ms，则理论应该发送6K，

2、计时部分

2.1 thisLoopTick 在while()中定义const DWORD thisLoopTick = timeGetTime();代码块常变量，初始化之后没再赋值

2.2 lastLoopTick

2.2.1RunInternal局部变量，初始化DWORD lastLoopTick = timeGetTime();

2.2.2

UINT UploadBandwidthThrottler::RunInternal()

{

DWORD lastLoopTick = timeGetTime();

……

While(doRun)

{

DWORD timeSinceLastLoop = timeGetTime() - lastLoopTick;

……..

const DWORD thisLoopTick = timeGetTime();

if(allowedDataRate != \_UI32\_MAX) //在有速度限制的情况下，

{ …….

else if(\_I64\_MAX/timeSinceLastLoop > allowedDataRate && \_I64\_MAX-allowedDataRate\*timeSinceLastLoop > realBytesToSpend)

{

if(timeSinceLastLoop > sleepTime + 2000) ///snow:超过2秒

{ ……

timeSinceLastLoop = sleepTime + 2000;

lastLoopTick = thisLoopTick - timeSinceLastLoop;

}

…..…

}

………..

}

lastLoopTick = thisLoopTick;

if(bytesToSpend >= 1 || allowedDataRate == 0)

{

发送数据 ，分四批发送，首先发送控制包信息，其次发送m\_StandardOrder\_list中长时间没发送过数据的socket，第三再正常发送m\_StandardOrder\_list中各socket，最后如果带宽没用完，就再次发送m\_StandardOrder\_list中的各socket，尽量用完带宽。如果实在用不完，则允许在下一循环中多发送999字节。

}

….

}

…….

}

2.3 在循环开始之前，记下当前时刻，存入lastLoopTick，作为初始值（其实有必要吗？）；每次循环开始执行，用循环开始时间，减去lastLoopTick，为运行已过时间，存入timeSinceLastLoop。因为第一次循环时，计算timeSinceLastLoop的语句紧挨着lastLoopTick赋值语句，所以timeSinceLastLoop=0。在循环执行中，当准备发送数据 的时候，记下当前时刻，用当前时刻的值去更新lastLoopTick，这样在第二次或以后的循环时，timeSinceLastLoop的值就相当于上次循环时发送数据的时间。当需要限速时(allowedDataRate != \_UI32\_MAX)，计算spentBytes/timeSinceLastLoop就可以获取当前上传速度了。

# UploadSpeedSense

automatically finds the best upload speed for your connection.

The goal is to make eMule work right out of the box, without need for configuration of upload speed. The users should be able to just leave the upload speed limit at default 0 (unlimited) in preferences and relax. If they use other programs that want bandwidth, UploadSpeedSense will automatically lower the upload limit for eMule while the other transfer is going on. When the transfer is done, UploadSpeedSense raises the upload limit back to normal speed. UploadSpeedSense will not work for multihomed hosts.

This version contain only UploadSpeedSense, and no other of the ZZ features. I've created this separate version to make it easier to evaluate this feature and to hopefully get it included in the official version.

UploadSpeedSense is based on the DynUp idea, and in fact uses a few lines of code from DynUp. Thanks!

## How UploadSpeedSense works

* Uses traceroute to find one of your ISP's routers to ping. The tracerouting finds the last common host for 10 randomly chosen ips in your server list and known clients. TTL is increased one step at a time, and each host is pinged until the returned ip differs. This means none of the hosts are actually pinged, only your ISPs infrastructure gets the pings. Last common host is saved, and one ip of the routers one hop furhter than that last common host (this is the ping ip, the ip that pings will be sent to). During this phase the lower right corner of eMule will say "Preparing...". When a good host to ping has been found, the ip of that host is reported in the debug log. There's currently no way to manually set which host should be pinged. This may be changed in a future version.
* UploadSpeedSense now regurarly pings the ping ip, but it sets the TTL one to short, to get the ip from the last common host back. It uses this ping to measure latency for the connection. When the ping gets to high, it lowers the upload limit. When the ping gets low, it raises the upload limit. You can see the current ping in the lower right corner in eMule (statusbar).
* If the responding ip for the set TTL changes (remember, we set TTL one shorter than the number of hops to the ip we ping), then UploadSpeedSense assumes that the topology has changed, and will redo the tracerouting phase to find the new topology. This may happen if the computer has been reconnected to the internet, i.e a modem redial.

Preferences

You might need to experiment with the defaults in the Extended settings page in preferences.

These are the defaults:

Ping Tolerance: 800%  
Up Slowness: 1000  
Down Slowness: 1000  
Max number of pings for average: 1

Higher ping tolerance will make it accept a higher ping before it lowers the speed.

The slowness values controls how fast/slow the speed is changed to match the ping. Higher values makes it change the speed slower. Lower values makes it change the speed faster. The higher values, the smoother graph, and the slower it reacts to ping changes.

"Max number of pings for average" controls how many pings it will average to calculate the ping. Higher value will make it react less, and slower, to ping changes. Too high value here will make the upload graph wave-formed.

## How to tweak

First set the upload limit value to the highest speed you want to allow eMule to upload to. UploadSpeedSense will never raise upload limit to a higher value than this value. If you want UploadSpeedSense to control speed completely, disable upload limit.

**The UploadSpeedSense are available in Preferences->Extended settings.**

**Find best upload limit automatically:** Check this to enable.

**Lowest allowed upload speed:** the lowest speed that UploadSpeedSense is allowed to set.

**Ping tolerance (%)**: This value is only used if "Method for ping tolerance" is set to "Percent (%)". Start at 500% and then try to raise it 100 or lower it 100 and see the difference. The lower this value, the lower upload speed you will get. 100% is the lowest value possible, and will probably make USS lower your upload limit to whatever MIN SPEED is set to.

**Ping tolerance (ms):** If you prefer, you can set the actual ping value you want. This value is only used if "Method for ping tolerance" is set to "Milliseconds (ms)".

**Going up/down slowness:** These values effect how fast the speed is adjusted. You will probably want both of these to both be set to one single value. Start these at 1000. If you get a roller coaster in the upload graph, try to raise them to 2000, then 3000, etc. The HIGHER the value, the SLOWER USS changes the upload limit. Too high value here will more or less prevent USS from changing upload speed at all.

**Number of pings for average:** Should ALWAYS be 1. I've never gotten good results with any other value.