

## Re: large tcp window

**Source:**

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.networks/2004-11/0220.html>

---

**From:** Arkady Frenkel (*arkadyf\_at\_hotmaildotx.com*)

**Date:** 11/10/04

Date: Wed, 10 Nov 2004 12:05:56 +0200

Additionally about TCP window size ( TcpWindowSize registry key ) on XP ( at least ) without 1323 option set :

If line >= 100 Mbps the size set in setsockopt() not important and TcpWindowSize ~ 64K ( 60480 ) set

If line = 10 Mbps if the size set in setsockopt() more than 17,520 ( default for 10 Mbps ) I see that in TCP packet,

if less than 17,520 used for TCP window size .

for TcpWindowSize in the registry that behave differently if it set it taken as a size for both : 10 or 100+

the same for TcpWindowSize in the registry of interface.

so I saw that on the start on connect with ACK win size is 20000 , next SYN its 20440

( for 30000 it take 30660 , and for 10000 – 10220) and when data send ( with PSH,ACK ) that 20440.

That exact behavior described in

[download.microsoft.com/download/7/7/1/7716a332-d3af-4ad5-b249-38ca97db023e/tcpip2000.doc](http://download.microsoft.com/download/7/7/1/7716a332-d3af-4ad5-b249-38ca97db023e/tcpip2000.doc)

When I set global TcpWindowSize to 30000 and local ( in the interface ) to 35000 I see that taken window size 30000

When I changed and set global TcpWindowSize to 35000 and that of interface to 30000 the 30000 was taken,

so the small one taken.

So some precedence described in "TCP Receive Window Size Calculation and Window Scaling (RFC 1323)"

of

<http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/networking/tcpip03.msp>  
work but not exactly like described , no such in tcpip2000.doc at all.

The precedence there :

The default advertised TCP receive-window size in Windows Server 2003 depends on the following, in order of precedence:

1.

The SO\_RCVBUF WinSock option for the connection.

2.

The per-interface TcpWindowSize registry setting.

3.

The global TcpWindowSize registry setting.

4.

Autodetermination based on the NDIS-reported bit rate of the media The stack also tunes itself based on the media speed:

. Below 1 Mbps: 8 KB

. 1 Mbps – 100 Mbps: 17 KB

. Greater than 100 Mbps: 64 KB

For 10 and 100 Mbps Ethernet, the window is normally set to 17,520 bytes (17 KB rounded up to twelve 1460-byte segments.) There are two methods for setting the receive window size to specific values:

. The TcpWindowSize registry parameter (see Appendix A)

. The setsockopt() Windows Sockets function (on a per-socket basis)

Arkady

"Arkady Frenkel" <arkadyf@hotmaildotxcom> wrote in message  
news:OcN5XstxEHA.1188@tk2msftngp13.phx.gbl...

> *Hi, Alun !*

>

> "Alun Jones [MSFT]" <alunj@online.microsoft.com> wrote in message  
> news:ObJEfzrxEHA.2192@TK2MSFTNGP14.phx.gbl...

>> "Arkady Frenkel" <arkadyf@hotmaildotxcom> wrote in message

>> news:e572hidxEHA.2620@TK2MSFTNGP10.phx.gbl...

>>> *You can't send packet more than MTU ( that 1500 bytes in ethernet).*

>>> *TcpWindowSize or sliding window is size of buffer peer can send*

*without*

>>> *waiting for ACK.*

>>

>> *Correct.*

Re: large tcp window

>>  
>>> *That way of TCP to increase throughput opposite to decreasing effect of*  
>>> *Nagle algorithm.*  
>>  
>> *Incorrect. The Nagle algorithm affects only data smaller than MTU in size –*  
>> *it cannot possibly have any effect on data over that size, and the point of*  
>> *setting large window scale sizes is so that you can send large amounts of*  
>> *data over a link that has a large latency time.*  
>>  
>  
> *I didn't mention the size of data sending , btw , but I mean a lot of small*  
> *buffers sending. That example ( Stevens/Snader ) of more effective using of*  
> *TCP over UDP even in such case where*  
> *on the first glance Nagle have to decrease performance but with TCP sliding*  
> *window that not happen and TCP can be more effective than UDP ( even with*  
> *Nagle :) )*  
>  
>> *Think back to the calculation – window size = bandwidth x latency.*  
>>  
>> *So, if you have a pipe that can take 10 Mb/s, and has a latency of a tenth*  
>> *of a second, its window size (the amount of data that needs to be sent*  
>> *unacknowledged to fill the pipe) will be  $10 \times 0.1 = 1$  Mb. A pipe that can*  
>> *take only 1 Mb/s but has a latency of one second will require the same*  
>> *window size. [I was first introduced to the concept of large window scale*  
>> *factors several years ago by someone who wanted to "communicate with*  
>> *orbiting space vehicles". It was difficult trying to replicate that*  
>> *customer's configuration in the lab :-)]*  
>>  
> *That correct , but 100 mbps net set 64K window ( unless I set different*  
> *size in registry ), without*  
> *calculations :)*   
> *Arkady*  
>  
>>> *So if you set support of 1323 you send the same size packet ( 1500 ),*  
> *but*  
>>> *not wait each time 200 ms ( by default ) in leu of Nagle algorithm*  
from  
>>> *waiting Ack*  
>>  
>> *Now that's just plain mumbo–jumbo.*  
>>

> > *Nagle isn't the only factor that causes TCP applications to have  
> > difficulties, and it isn't even the biggest issue. It's actually a  
> > non-issue, almost all of the time. And it's the interaction between  
Nagle  
> > and delayed ACK that causes a problem for poorly-written applications  
and  
> > protocols. If disabling Nagle helps your application, reconsider how  
you  
> > use TCP, because you're doing it wrong. Nagle is there to protect the  
> > network from poor-performing apps, and to give you more of your  
bandwidth  
> > in  
> > data, rather than in headers.  
> >  
> > Nagle cannot possibly bear on anything to do with large window scale  
> > factors, because Nagle affects low-usage scenarios, and LWS indicates  
> > high-usage.  
> >  
> > Don't paint Nagle as a problem – it's a *\_solution\_*. Particularly,  
please  
> > don't paint Nagle as a problem when it can't possibly have any effect.  
> >  
> > *Alun.*  
> > ~~~~  
> >  
> >  
>  
>*