

Re: recv WSAENOBUFFS error

Thanks again,
Mason

On Dec 19, 2:43 am, "Arkady Frenkel" <arka...@xxxxxxxxxxxxxxxxxxxx>
wrote:

Additionally.
You can set kernel-mode buffer with setsockopt but you'll
quickly finish
your non-page (kernel) memory if you'll start to do it. For
large
buffers
you can use zero-buffering (set size of kernel socket buffer
to 0 with
setsockopt). In this case winsock will use your user-mode
buffer which
can
be very big.
Arkady
P.S. BTW default kernel-mode buffer size for socket is 16K
(from W2K)

<i...@xxxxxxxxxxxx> wrote in
message:news:1166504011.731450.124570@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Hi,

If I understood you correctly, you are trying
to send large data chunks
in one command or atleast with big pieces of
data (more then 1.5kper
call)

Internet packets are 1.5k long, so it's advised
to send data over TCP
using 1.3k-1.5k (I advise 1.3k) data length,
and to send the expected
data size (like 64mb in your example) before
sending the actual data,

Re: recv WSAENOBUFFS error

the reason is that TCP is stream oriented and not packet oriented (unlike UDP which is what you send is what you get), so the other side will not receive the data in the same exact way you sent it, what I mean is that recv won't have to look exactly like your send, you can send 1.5k then 1.5k and the user will get 2k then 1k.

When sending data bigger than 1.5k it will cause the packet to fragment using IP fragmentation option and will cause delays in send/recv and every packet loss/retransmission will cost you in greater bandwidth and loss of performance.

Good luck,
Barak Weichselbaum,
<http://www.komodiam.com>

boarding_ga...@xxxxxxxxxxxx wrote:

Hi,

I don't have a lot of experience with sockets, but I'm debugging a problem in some existing software, and could use some help.

I have a couple of processes talking through a non-blocking socket. In general, everything works fine, until a particularly large data transfer (~64 MB) is

Re: recv WSAENOBUFFS error

attempted. When the receiving side tries to do a recv, a WSAENOBUFFS error is returned. I've been searching around a lot, and have found a workaround to this problem, but I want to make sure it's an appropriate fix, and would like to understand exactly why it fixes the problem, and what impact it might have on performance.

The original code determines the length of the data (contained in the

first several bytes of the data), then allocates the appropriate space for the data, and calls recv with the full length as the "len" parameter. This recv call fails when that large data length is encountered. Based on information I found during my research, I changed

this code to send 64K as the maximum length to recv, and I loop and piece together the results until all of the data is received. This seems to "fix" my problem. However, I've also seen suggestions about changing the receive buffer size using setsockopt,

Re: recv WSAENOBUFFS error

changing various registry settings, etc (none of which have seemed to help me). What I would like to understand is what the various buffers involved in the socket transfer are, and how they can be configured. It seems like the

parameter I'm changing has some effect on some sort of memory allocation somewhere, but I'm not sure what it's actually changing. Also, I have concerns about performance impact. It seems like when I pass large amounts of data (< 64MB, so it succeeds), multiple recv calls are required to get all of the data anyway, so I'm hoping that this change won't really affect the performance too much.

Any advice or clarification would be greatly appreciated!

Thanks for your time,
Mason– Hide quoted text
-- Show quoted text --
Hide quoted text -- Show
quoted text –