

Re: Direct Copying To Share Memory In NDIS ProtocolReceive

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.development.device.drivers/2007-03/msg00021.html>

- *From:* "Thomas F. Divine" <tdivine@NOpcausaSPAM>
 - *Date:* Thu, 1 Mar 2007 14:17:14 -0500
-

Best approach is still to use standard DeviceIocontrol or Read instead of shared memory. The effect you seek is achievable using this method.

Use Read with DO_DIRECT_IO. Application allocates large structured buffer with room for several (many) packets. Driver uses MmGetSystemAddressForMdlSafe to map buffer to kernel memory. Then pends the read and stashes buffer and IRP internally. Driver fills buffer with multiple packets and eventually completes the read IRP.

This minimizes user-kernel transitions (i.e., number of I/O calls) – which is a performance killer. Also, there is no memory copy between user-mode buffer and kernel-mode buffer (because DO_DIRECT_IO was used). In addition, driver is safely informed (via Cleanup/Close callbacks) when application leaves the building.

Improvements on this include 1.) having multiple concurrent Reads in progress so driver already has another Read buffer queued when current one becomes filled and 2.) having a timer that will complete a Read eventually even if buffer is not filled.

This, or variations on this, should work for you without having to have shared memory and events that are not part of the standard I/O scheme.

One performance killer is user-kernel transitions. This would be the number of I/O calls. Or, almost exactly the same bad effect, the number of times user-mode must wait on event set by driver.

In broad terms it is often not the amount of data that is transferred from driver to user-mode that is a performance issue. Instead, the performance bottleneck may be simply the number of events that must be used to synchronize the transfer.

Hope this helps.

Thomas F. Divine

"Le Chaud Lapin" <jaibuduvn@xxxxxxxxxx> wrote in message
<news:1172728791.133314.195370@xx>

I have an NDIS driver based on NDISPROT.

I am wondering if I can obviate much of the NDIS packet allocation by copying indicated packets in ProtocolReceive directly to user-mode

Re: Direct Copying To Share Memory In NDIS ProtocolReceive

+kernel-mode shared memory section. I would call KeWaitForSingleObject in ProtocolReceive on a semaphore to wait (0 seconds) for one of, say, 64 fixed-size packet (frame) slots in the shared section, and copy the packet directly, taking note which slots were filled. If the wait breaks because there are no slots, then I would simply return NDIS_STATUS_NOT_ACCEPTED. When ReadFile is called, I would complete the IRP by receiving not the packet data, but an array of indexes showing which slots were filled in.

I was wondering two things:

1. What is the computational cost of KeWaitForSingleObject with timeout of 0? (relatively of course)
2. What kind of performance improvement can I expect using this method, if any.

I was trying to come up with a way of batching receives, and this seemed reasonable.

-Le Chaud Lapin-