close window

e-Newsletter Exclusive

Print

Tuning AIX Network Performance

January 2014 | by Jaqui Lynch

By default, the network tunables on an AIX system aren't set optimally for anything above a 100 MB network. Given that most customers are running GB or 10 GB networks, the first step to improving network performance is to set some basic tunables. The first set is done using the no command.

```
Tunables
no   -p -o rfc1323=1
no   -p -o tcp_sendspace=262144
no   -p -o tcp_recvspace=262144
no   -p -o udp_sendspace=65536
no   -p -o udp_recvspace=655360
```

This example sets TCP send and receive buffers to 256K, UDP send to 64K and UDP receive to 640K. It's common to see the TCP values changed but many leave UDP at the defaults. Since DNS and other protocols use UDP, it's important to increase those values. Typically, you receive about 10 times as many UDP packets as you send, hence the difference in the values.

These defaults only come into play if nothing has been set on the actual adapter, which is the case for certain (more recent) AIX releases. Check by running the "ifconfig –a" command. You'll see something like the following:

```
en0: flags=1e080863,480
```

If the TCP send, receive and/or rfc1323 is set, they should be changed to match the above, unless the settings on the adapter are larger. The adapter can be set as follows:

```
chdev -l en0 -a tcp_recvspace=262144 –a tcp_sendspace=262144 –a rfc1323=1 –P
```

Depending on the load (I do this for the base adapters on my SEAs but also on busy systems), you may want to increase the adapter transmit queues. For Gbit adapters:

```
chdev -l ent0 -a txdesc_que_sz=1024 -a tx_que_sz=16384 -P
```

Both changes would be activated after a reboot (the –P sets that).

What are these settings?

- tcp_recvspace specifies how many bytes of data the receiving system can buffer in the kernel on the receiving sockets queue.
- tcp_sendspace specifies how much data the sending application can buffer in the kernel before the application is blocked on a send call. The recommendation for performance is that it should be set to at least the same size as tcp_recvspace. For high-speed adapters, it should be at least twice the size of tcp_recvspace.

- rfc1323 is also known as the TCP window scaling option. This must be set to 1 on both sides of the connection otherwise the effective value of the tcp_recvspace tunable will be 65536, even though you may have set it to 262144. The default is rfc1323=0 (off) so it's important to set this tunable if you plan to set TCP send and receive higher than 65536 (which I'm recommending).
- udp_sendspace is used for UDP datagram buffering for send, and udp_recvspace controls the amount of space used for queuing incoming data on the UDP socket. Once the udp_recvspace limit is reached, incoming packets are discarded. You can tell if packets are being dropped by issuing the "netstat –p udp" command and looking for socket buffer overflows. udp_recvspace is typically set much higher than udp_sendspace as multiple datagrams can arrive at the same time and multiple applications tend to share sockets listening for UDP datagrams.
- tcp_nodelay is often used in a database environment. Setting this to 1, instead of the default 0, causes TCP to send each packet out immediately for each application send or write. Normally, TCP implements delayed acknowledgements, where it tries to piggyback a TCP acknowledgement onto a response packet; this delay is usually up to 200ms. The nagle algorithm means that a TCP connection can only have one outstanding acknowledgement for a small segment. Clearly this causes delays in sending further packets until either the acknowledgement is received or TCP can bundle up more data into a full segment. Setting tcp_nodelay to 1 is a dynamic change and a tradeoff between more network traffic versus better response time.

Looking for Problems

Several commands are useful when looking at network performance. nmon provides great statistics; the –O flag now provides network statistics on the SEA, which is very useful. Start with "netstat –v". Look for "S/W transmit queue overflow"—if you see these or "packets dropped due to memory allocation failure", you must increase the adapter transmit queue. You can check what it's set to using "lsattr –EL ent?".

Additionally you should look for receive or transmit errors, DMA overruns and DMA underruns. All are indicators of problems. And don't forget to check errpt as many problems show up there.

Virtualized Environment

When using virtual Ethernet or SEAs, you should also check the output from "netstat –v" for resource errors. If you see numbers beside "No resource errors:" for the adapter, scroll down to the "Virtual I/O Ethernet statistics" or "Virtual Trunk Statistics" and look for numbers in the hypervisor send or receive failures (see Example A). Typically these will match (or be close to) what you're seeing under "no resource errors". If this is the case, you'll need to tune some buffers. Scrolling a little further down in the netstat output, you'll find a section headed "Receive Information" followed by "Receive Buffers" (see Example B). You'll see five types of buffers—tiny, small, medium, large and huge. "Max Allocated" represents the maximum number of buffers ever allocated. "Min Buffers" is the number of pre-allocated buffers. "Max Buffers" is an absolute threshold for how many buffers can be allocated.

Compare the "Max Buffers" value for each buffer type to the "Max Allocated" number. If they're equal, increase the problem buffer by using the chdev command on the virtual Ethernet, not the physical adapter. The following commands increase the minimum and maximum for the small buffers. should be replaced by the actual virtual Ethernet ent.

```
chdev -l  -a max_buf_small=4096 -P

chdev -l  -a min_buf_small=2048 -P
```

This will ensure there's enough memory to vent buffer space quickly for the workload.

Just a Sample

This is just a small sample of some of the network tuning that can be done. Another option includes different MTU sizes (9000 or jumbo frames), however, these must be done in close coordination with the network team. The settings I provided should be useful as starting points to set parameters correctly and look for problems. Make sure you always test new settings on test servers first.

References

For more information on adapter settings, see page 247 of the Performance Management documentation

Example A

< Return to main article

Print Email

SEA

Transmit Statistics:	**Receive Statistics:**
Packets: 83329901816	Packets: 83491933633
Bytes: 87482716994025	Bytes: 87620268594031
Interrupts: 0	Interrupts: 18848013287
Transmit Errors: 0	Receive Errors: 0
Packets Dropped: 0	**Packets Dropped: 67836309**
	Bad Packets: 0
Max Packets on S/W Transmit Queue: 374	
Max Packets on S/W Transmit Queue: 374	
Current S/W+H/W Transmit Queue Length: 0	
Elapsed Time: 0 days 0 hours 0 minutes 0 seconds	
Broadcast Packets: 1077222	Broadcast Packets: 1075746
Multicast Packets: 3194318	Multicast Packets: 3194313
No Carrier Sense: 0	No Carrier Sense: 0
DMA Underrun: 0	DMA Overrun: 0
Lost CTS Errors: 0	Alignment Errors: 0
Max Collision Errors: 0	**No Resource Errors: 67836309**
Virtual I/O Ethernet Adapter (l-lan) Specific Statistics:	
Hypervisor Send Failures: 4043136	
Receiver Failures: 4043136	
Send Errors: 0	
Hypervisor Receive Failures: 67836309	

< Return to main article

Example B

< Return to main article

Print Email

Virtual Trunk Statistics

Receive Information

Receive Buffers

Buffer Type	Tiny	Tiny	Medium	Large	Huge
Min Buffers	512	512	128	24	24
Max Buffers	2048	**2048**	256	64	64
Allocated	513	2042	128	24	24
Registered	511	506	128	24	24
History					
Max Allocated	532	**2048**	128	24	24
Lowest Registered	502	354	128	24	24

< Return to main article