Care and Feeding for Better VM TCP/IP Performance

August 1998

Bill Bitner
IBM Endicott
1701 North St.
Endicott, NY 13760
607-752-6022
bitner@vnet.ibm.com

Romney White
IBM Endicott
1701 North St.
Endicott, NY 13760
607-755-8276
romney@vnet.ibm.com

Disclaimer

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

Any performance data contained in this document was determined in a controlled environment and, therefore, the results which may be obtained in other operating environments may vary significantly.

Users of this document should verify the applicable data for their specific environments.

It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country or not yet announced by IBM. Such references or information should not be construed to mean that IBM intends to announce such IBM products, programming, or services.

Should the speaker start getting too silly, IBM will deny any knowledge of his association with the corporation.

Trademarks

The following are trademarks of the IBM Corporation:

- IBM
- OfficeVision
- VM/ESA
- ACF/VTAM

Sources of Information

- TCP/IP Tutorial and Technical Overview Redbook, GG24-3376.
- IBM TCP/IP Performance Tuning Guide, SC31-7188
  - Also contains OS/2, DOS, and VM information.
  - Concepts, Tuning Information, and benchmark data.
- VM TCP/IP Home Page
  - http://www.vm.ibm.com/related/tcpip/
- IBM TCP/IP List
  - IBMTCP-L@VM.MARIST.EDU
  - Newsgroup: bit.listserv.ibmtcp-l
- Talklink TCPIP CFORUM
- TCP/IP Function Level 310 Planning and Customization, SC24-5847-00
- TCP/IP Function Level 310 User's Guide SC24-5848-00

A good overall book for TCP/IP concepts is the first redbook mentioned above.

There is little VM TCP/IP performance information in the VM libraries, but there is a manual which does cover a lot of good performance related information. That is the IBM TCP/IP Performance Tuning Guide (SC31-7188). The PTG contains good background material on TCP/IP concepts that relate to performance and then show application of them to MVS, VM, OS/2, and other environments.

When looking for TCP/IP performance information, why not check out the internet. The VM TCP/IP home page URL is given in the foil. This presentation will be made available there and updated as it is improved. In addition, performance related questions can be asked on various lists on the net.

Agenda

- Will not cover all of TCP/IP Configuration
- Highlights from a biased (performance) view
  - Performance related knobs
  - Key changes in function level 310
  - Key server machines
- TCP/IP Function Level 310 assumed

This presentation is not meant to cover everything there is to cover about configuring TCP/IP. It will look at some of the most important things from a performance perspective. We will only look at a subset of the TCP/IP functions and applications. We will concentrate on the TCP/IP function level 310 which is a feature of VM/ESA 2.3.0.

Where Does What Get Set?

- DTCPARMS files
- TCPIP DATA
  - 14 Knobs, not really performance
  - Configuration information
- OBEYFILE
  - Used in making dynamic changes
- PROFILE TCPIP or Configuration File

The first question I had when starting to look at this is where are the knobs? Are they commands or settings in setup files? The first file DTCPARMS contains pointers to other files and describe which virtual machines are responsible for which application. The TCPIP DATA file has additional configuration information and is used by end users as well. There is little here that affects performance, unless you count the fact that things do not perform at all if this is incorrect. The OBEYFILE can be important for making dynamic changes, in case you want to experiment with the servers up or to correct mistakes made. We will spend quite a bit of time in the PROFILE TCPIP or configuration file for the Stack.

TCP/IP Config: 47 knobs?

- 15 for Storage Pool Sizes (perf)
- 3 for time thresholds (perf)
- 2 for initialization/start up
- 10 for configuration (perf)
- 10 for Trace and Friends (perf?)
- 1 for Monitor (perf)
- 6 for control

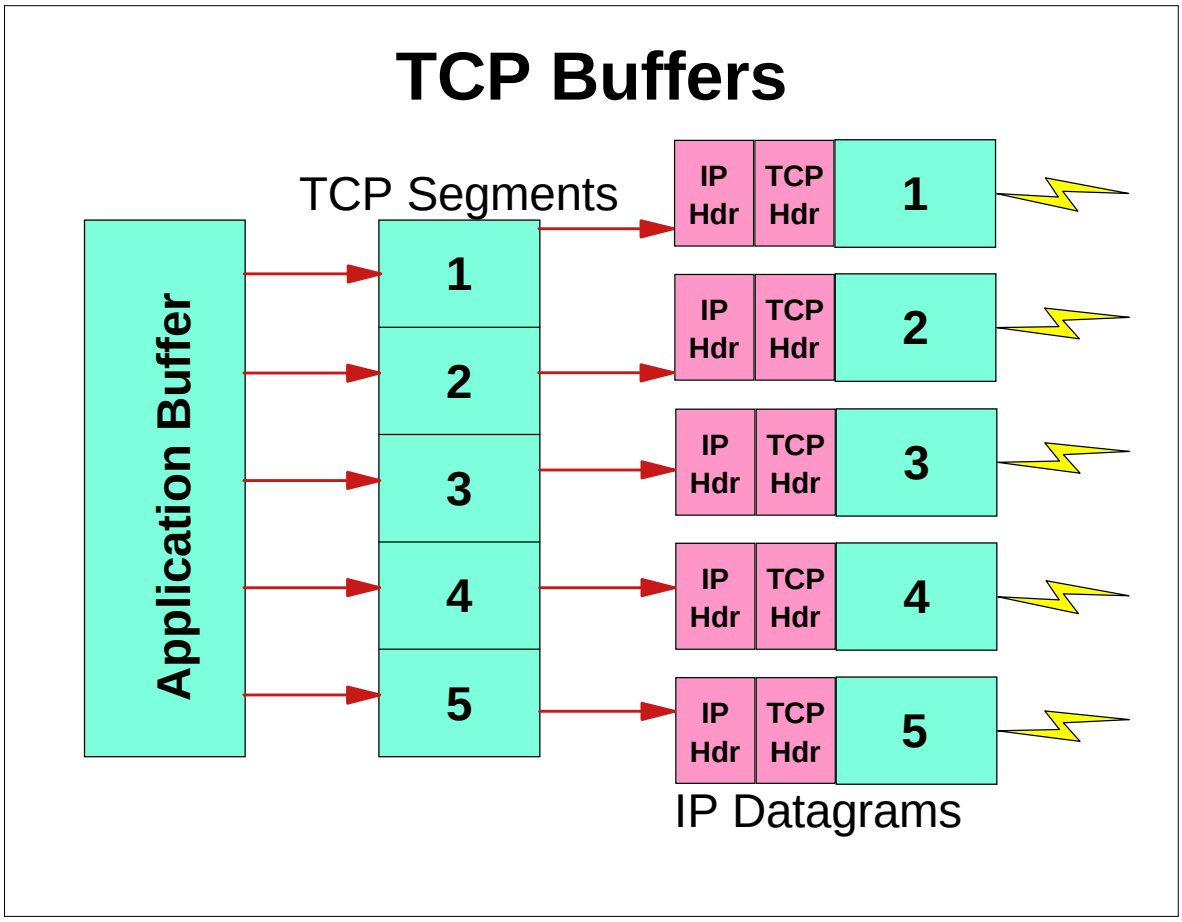
I broke the TCPIP PROFILE or config file up into 7 groups. The 6 in the control group are not performance related. The startup statements are not performance sensitive either. The 10 for tracing and debug could be considered performance in terms of slowing things down. The performance related ones involve storage structures, time thresholds, configuration, and monitor.

Buffers, Segments, Datagrams, Windows, and MTU Sizes

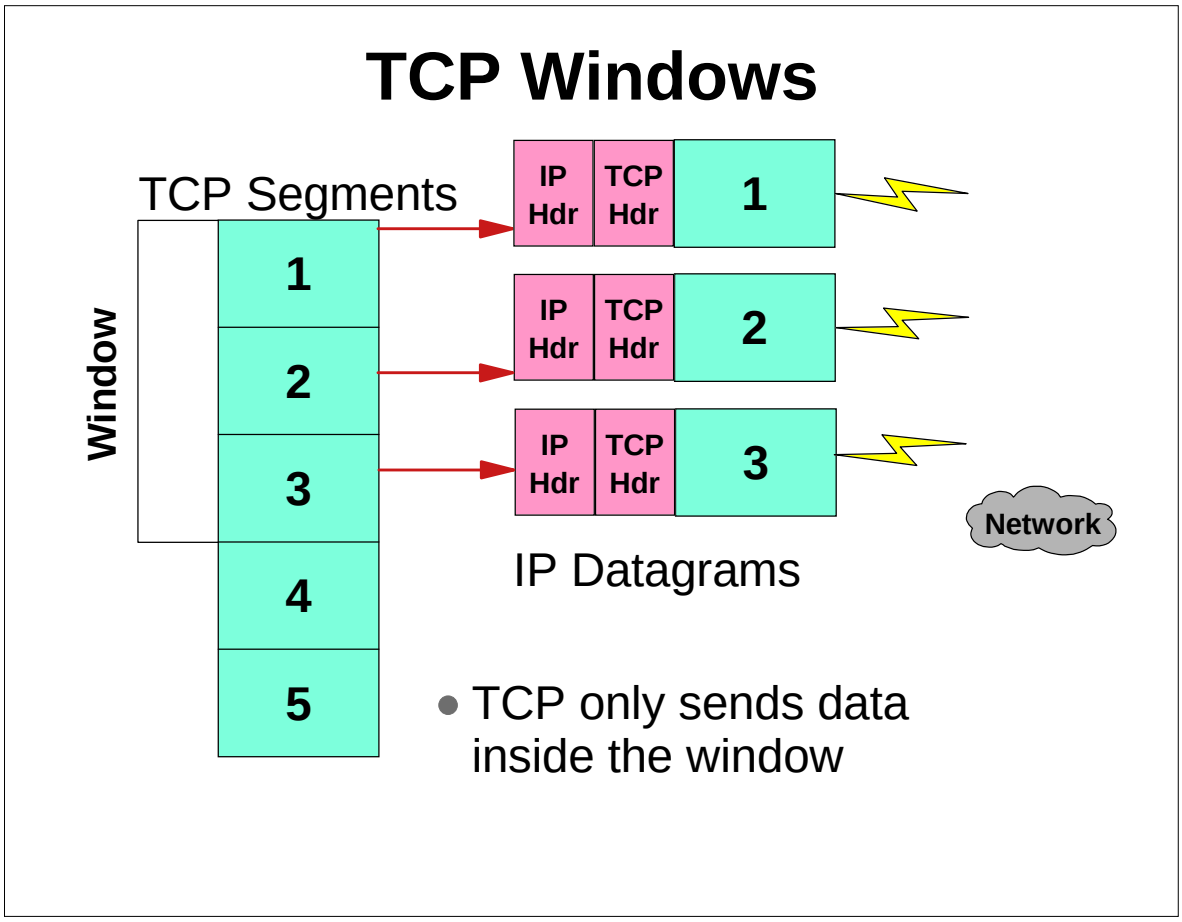
- Two main Protocols:
  - TCP = Transmission Control Protocol
  - UDP = User Datagram Protocol
- Both user flavors of send and receive buffering.
- TCP is connection-oriented and provides control, sequencing, and recovery
- UDP just sends and receives buffers with no interest in whether they arrive

It might be good to step back and discuss some concepts.

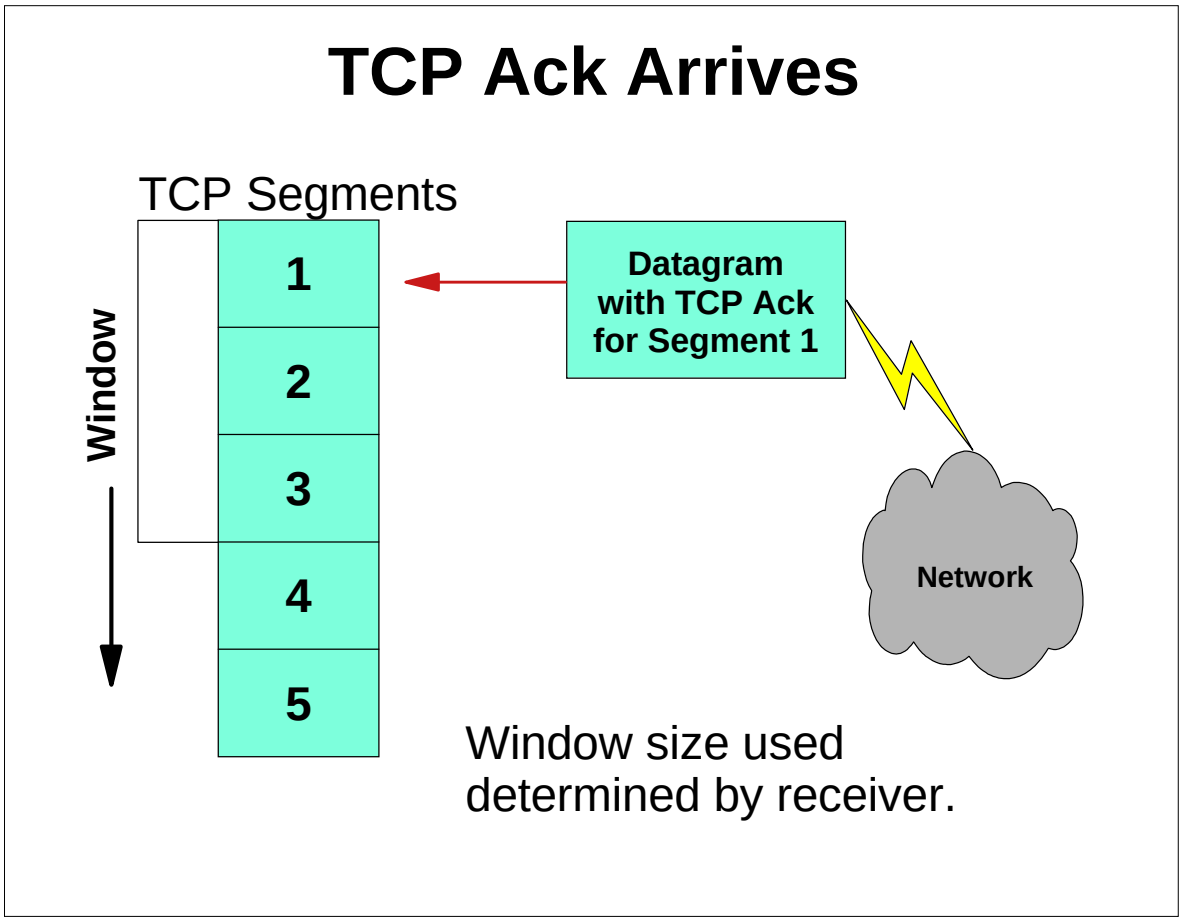
There are several protocols associated with TCP/IP, but two main ones: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Both use send and receive buffering. However, TCP is based on connections while UDP is not. TCP also uses acknowledgments to deal with unreliability in the network and the IP layer. UDP trusts the network.



The diagram here shows how TCP uses buffers. The application data is split into segments by TCP. These segments, with TCP and IP headers attached, are IP Datagrams and are sent over the network.

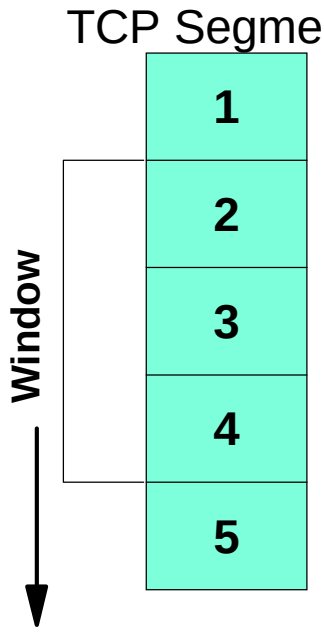


TCP uses windowing, both on the sending and receiving sides. In our example, TCP has a window of 3 segments. It will only send this amount of data until acknowledgments start to come back.



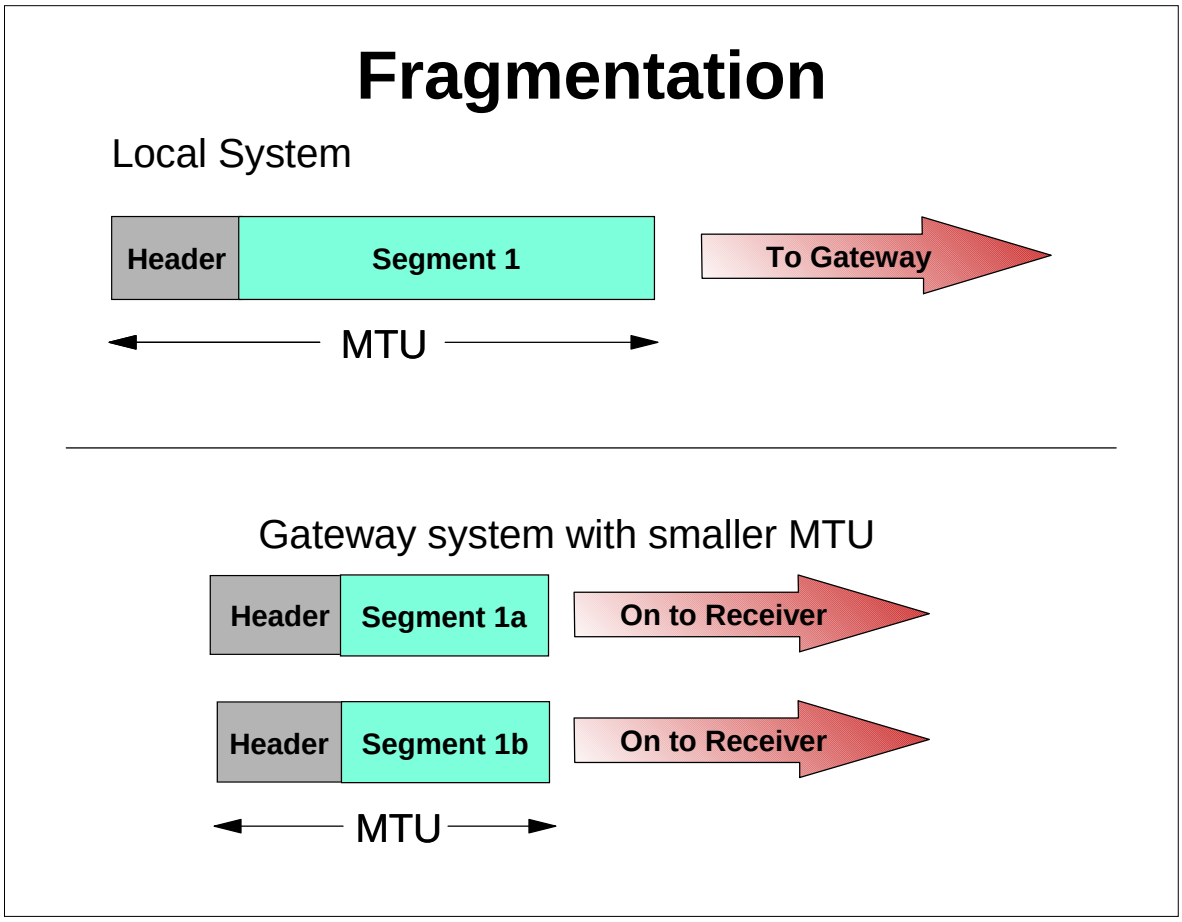
After the receiving system delivers the data to its client, it will send an Acknowledgment back. At this point, the sender can slide the window forward. The size of the window is actually controlled by the receiver, since it needs to have enough storage to hold that amount of data so it doesn't lose any.

TCP Window Slides



- The sending window slides to exclude segment 1 after the Ack for the last byte of segment 1 is received.
- A larger window means more data can be started down the pipe before waiting for an Ack.
- A larger window means more data will have to be retransmitted if there is a problem.

In sliding the window, it essentially stays the same size, but covers different segments. It is possible to receive an Ack before the entire window is filled. A larger window means more data can be kept in the pipe before waiting for an Ack. However, this also means that if there are failures, the Stack will need to retransmit a larger amount of data. And, any unacknowledged data occupies stack buffer space.

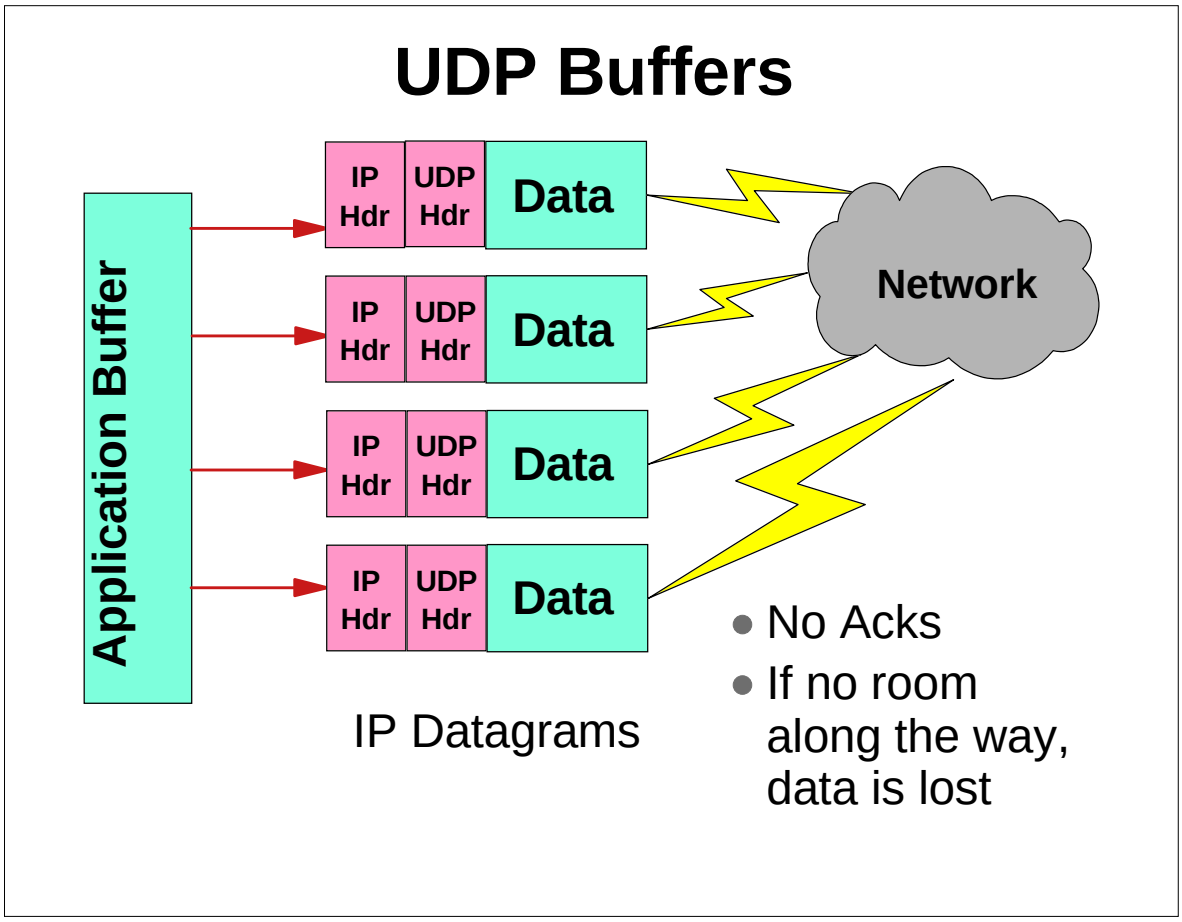


Fragmentation is another concept that is important to understand. Since not all points are directly connected, data will often need to pass through gateways or intermediate nodes. MTU is the maximum transmission unit. If an intermediate node or gateway has a smaller MTU size, then the data needs to be split up. Data is always sent with header information attached. The final destination would be responsible for putting all the fragments together again.

Cost of Fragmentation

- Overhead to fragment into smaller packets
- More packets to send
- Higher header to data ratio
- Overhead to defragment at the receiver
- Larger MTU size reduces stack overhead
  - Cost at the packet level, not the byte level
  - May be constrained by the network
  - May be able to use different MTU sizes to different hosts on same network

There is overhead to fragment the data into smaller packets, as well as to send the additional packets that result. Fragmentation can occur multiple times if various gateways along the way have subsequently smaller MTU sizes. Since header information is added with each fragmentation, the ratio of header data to application data increases, lowering efficiency. There will also be overhead at the receiving end to defragment the data.



UDP also uses buffers. However, windows are not used. There are no acknowledgments passed back. If somewhere along the way, a receiver or gateway cannot keep up, the data can be quietly discarded.

Profile: Storage Knobs

- Overestimating these is much less of a problem in 310
- Monitor with
  - NETSTAT POOLSIZE command
  - monitor data

Overestimating these storage knobs used to greatly increase the virtual storage requirements and also paging. In function level 310 this is much less of a problem, but you still need the virtual storage. You can use monitor data or the NETSTAT POOLSIZE command to check these pools. An example is shown in the following foil.

```
netstat poolsize
VM TCP/IP Netstat Level 310
TCPIP Free pool status:
Object       # alloc     # free   Lo-water Permit size
======       =======     ======   ======== ===========
ACB            5000        4933     4643         500
CCB             750         582      582          50
Dat buf        1200        1159     1131         240
Sm dat buf     5000        4796     4655         500
Tiny dat buf      0           0        0           1
Env            1250        1250     1143         125
Lrg env          75          74       71          15
RCB              50          50       50           3
SCB            2000        1941     1667         133
SKCB            256         219      207          17
TCB            5000        4743     4451         333
UCB             500         487      480          33
Add Xlate      1500        1489        1           5
IP Route       3000        2992        1          60
Ready;
```

Here is a sample NETSTAT POOLSIZE command response. For each pool, the number of blocks in the pool, the number available, the minimum number available since initialization, and the level at which the stack starts to get nervous are shown.

The last two lines illustrate a bug as the Lo-water mark field is always reported as "1". This will be fixed.

Profile: Control Blocks

- See Planning and Customization manual for usage notes
- ACB, ADDRESSTRANSLATION, CCB, IPROUTE, RCB, SCB, SKCB, TCB, UCB
- Too few typically means something will stop working.
- Too many impacts virtual storage requirements

The Planning and Customization manual has numerous usage notes for the various control block poolsize settings. Having too few of these will typically mean something stops working. While having too many can increase virtual storage requirements but allows for future growth.

Profile: TCP Related

- DATABUFFERPOOLSIZE - number and size of buffers
- SMALLDATABUFFERPOOLSIZE - number of 2048 byte buffers
  - Telnet will use if defined
- TINYDATABUFFERPOOLSIZE - number of 256 byte buffers
  - used for small exchanges (offload)

There are three storage pool size settings associated with TCP. The DataBufferPoolSize controls the number and size of buffers for most TCP activity. SmallDataBufferPoolSize controls the number of 2K buffers, which Telnet will use if available. TinyDataBufferPoolSize controls the number of 256-byte buffers, which are used for the 3172 Offload feature and the Telnet session connection exit.

Profile: Window Size

- Prior to 310 window size is
  - a constant multiple of the data buffer size
- In 310 window size is the product of
  - Data buffer size
  - values in new DATABUFFERLIMITS statement
- DATABUFFERLIMITS
    - used to limit window size for RFC 1323
    - receive and send values

In TCP/IP 2.4 and earlier releases the window size was based on a hardcoded multiple of the data buffer size. In TCP/IP function level 310, the maximum window size is the product of the data buffer size and values (receive and send) in the new DataBufferLimits statement. This was implemented with the support for RFC 1323 (long fat network).

Profile: IP Related

- ENVELOPEPOOLSIZE - control number of buffers for datagrams of 2048 bytes or less
- LARGEENVELOPEPOOLSIZE - control number and size
  - datagrams larger than 2048 bytes
  - staging for defragmenting
- The Stack tries to use best sized envelope

IP-related storage knobs include EnvelopePoolSize and LargeEnvelopePoolSize. The regular envelopes handle datagrams of 2048 bytes or less. You determine the size of the large envelopes which are used for datagrams greater than 2048 bytes and for defragmention.

Profile: Initialization

- ASSORTEDPARMS
  - New: CHECKCONSISTENCY restores the original checking of control block structures
  - New: NORFC1323 disables initiation of support for long fat networks.

The AssortedParms statement has two new options on it that are performance-related. The first, CheckConsistency, turns on the checking of control block structures. This validation processing was made optional (with the default being OFF) in function level 310 and helps avoid unnecessary page references. The other option is NoRFC1323 which disables the initiation of support for long fat networks. TCP will still use window scaling, but only if it is initiated by the other end of a connection.

Profile: Configuration

- Maximum Transmission Unit (MTU) controlled in
  - GATEWAY statement
  - BSDROUTING statement if using ROUTED
- DEVICE/LINK statements have device specific values
- SNMP information
  - SYSCONTACT
  - SYSLOCATION

Configuration statements unrelated to performance include: HOME, PRIMARYINTERFACE, START, STOP, TRANSLATE. The performance related ones include Gateway and BSDRouting, which control the MTU size for the respective connections. See the Planning and Customization manual for information on Device/Link statements. There are a number of device specific values that can be set.

Also if you are collecting SNMP information SysContact and SysLocation statements control some of the information that is retrieved through SNMP (Simple Network Management Protocol).

Profile: Monitor

- **MONITORRECORDS - controls creation of APPLDATA records**
  - Should be right after poolsize statements
  - Cannot change dynamically

TYPE	Records Produced
ALLRECORDS	all records
CPU	CPU consumption
CLIENT	client activity
HOMES	home address configuration
LINKS	link configuration
MIB	management information block
MOSTRECORDS	all except Scheduler (default value)
POOLS	storage pool configuration and use
SCHEDULE	Scheduler activity
TCP-SESSIONS	TCP session activity
TREES	hash tree activity
UDP-SESSIONS	UDP session activity

TCP/IP function level 310 creates CP monitor in the Appldata domain. The MonitorRecords statement in the profile file controls what type of data is collected. In order to get the most information, the MonitorRecords statement should follow the PoolSize statements immediately. The values selected here cannot be changed dynamically via an obeyfile. The default is MostRecords if no operand is specified.

You will also need to include APPLMON in the OPTION statement of the stack user's CP directory entry.

Profile: Time Thresholds

- ARPAGE
- INTERNALCLIENT parameters
  - INACTIVE - close inactive sessions
  - TIMEMARK - testing for client still there
  - SCANINTERVAL - how often to do above
- KEEPALIVEOPTIONS

TCP/IP will do checking of various functions based on intervals. Some of these intervals can be changed with the statements shown here.

Domain Name Server

- Typical approach in VM is to have a caching-only server and then go externally to an authoritative server when needed.
- NSMAIN DATA file
  - controls caching and what to cache

```
SMSG nameserv STATS
Ready;
    NS start time:             Thu Feb 12 03:30:30
    -------------------------------------------
    Total number of queries:  123
    Answers from cache:       31   (25%)
    Size of cache:            400  used: 4
```

The typical VM configuration has a caching-only name server which goes to an external authoritative name server when needed, though it is possible to set up an authoritative DNS in VM. The type of data cached can be controlled in the NSMAIN DATA file. You might want to cache the results of resolutions that fail, as well as those that are successful. You can check on some of the caching statistics by issuing a SMSG to the name server.

SMTP Server

- New in 310, SMTP uses *SPL so insure user directory has IUCV *SPL
- SMTP CONFIG file
  - LOG/NOLOG - use logging to disk for performance
  - MAXMAILBYTES - can protect from huge files
  - Several controls over waiting for connections and replies
- Configure so SMTP A-disk is on high performance DASD where possible

The SMTP (Simple Mail Transfer Protocol) was greatly enhanced for performance in function level 310. One of the design changes was to use the *SPL system service. Therefore, you are going to need to include IUCV *SPL in the SMTP machine's user directory entry. The SMTP CONFIG file controls various things for SMTP. You can disable logging altogether or change the target (the default is to minidisk, which is faster than spool). You can also control how much data can come in or go through SMTP in a single piece of mail with the MaxMailBytes statement. For best performance, configure SMTP with the A-disk on good performing DASD (e.g., DASD fast write).

Other Tuning Guidelines

- Typical SVM Tuning
  - QUICKDSP ON
  - Increase Share value
  - Reserve pages if necessary
- Match to use/workload
  - FTP likes bigger buffers
  - TELNET needs more smaller buffers
- Do not forget the other end of the communication if you control it

The traditional tuning we have done for years for server virtual machines applies to TCP/IP as well. This includes making sure it has access to processor and storage resources and never waits in the eligible list.

While larger buffers may help an FTP environment where lots of data is being pushed through the stack, it will not significantly help TELNET environments.
Also remember to look at the other end of the communication where possible for appropriate MTU size, RFC1323 use, and window sizes.

Helpful Utilities

- NETSTAT
  - displays information about the status of the local host
  - improved in 310
- PING
  - determines the accessibility of a foreign node
- TRACERTE
  - debug network problems
  - measures time between hops
  - userid must be in OBEY statement

Here are three useful utilities for debugging TCP/IP performance problems. The NETSTAT command will display various types of information. Some deficiencies were corrected in 310 such as making the interval display being scrollable and adding selectivity for various displays.

PING is a great way to quickly check network connectivity and responsiveness. By using various size data transfers with it, you can sometimes determine other interesting information.

The last one, TRACERTE, requires the user to be in the OBEY list. TRACERTE uses a series of Time-To-Live values to determine the time it takes packets to travel between hops in the network.

Summary

- Large number of buttons to push and knobs to turn
- Only a few are performance-related.
- Utilities exist (and are improving) to measure TCP/IP performance

As you have seen, there are a large number of configuration options in TCP/IP, but only a subset of them are performance related. Understanding some of the concepts behind how TCP/IP works can help you determine which tuning option to try. There are also a number of utilities, which continue to improve, to help you measure and debug performance of TCP/IP.
As always, thank you.