IBM VSE/ESA TCP/IP Performance Considerations

Wolfgang Kraemer

VSE Product Mgmnt
Dept 3221
71032-14 Boeblingen

WKRAEMER at DEVM
wkraemer at de.ibm.com

Update 2001-07-15

Contents

Contents

Contents

General Remarks

PART A.

General Remarks

Notes

Notes

This document is unclassified and intended for VSE customers.

All information contained in this document has been collected and is presented based on the current status.

It is intended and required to update the performance information in this document.

It is the responsibility of any user of this VSE/ESA document
- to use the latest update of this document
- to use this performance data appropriately.

The following documents are available via the INTERNET VSE/ESA home page

http://www.ibm.com/servers/eserver/zseries/os/vse

(http://www.ibm.com/s390/vse/ former URL)

Starting with VSE/ESA 2.4, these documents are also available on the VSE/ESA CD-ROM kit SK2T-0060 in Adobe Reader format (.PDF):

'IBM VSE/ESA 1.3/1.4 Performance Considerations'
'IBM VSE/ESA V2 Performance Considerations'
'IBM VSE/ESA Turbo Dispatcher Performance'
'IBM VSE/ESA I/O Subsystem Performance Considerations'
'IBM VSE/ESA VM Guest Performance Considerations'
'IBM VSE/ESA Hints for Performance Activities'
'IBM VSE/ESA TCP/IP Performance Considerations' (this doc)
'IBM DFSORT/VSE Performance Considerations'
'IBM VSE/ESA CICS Transaction Server Performance'
'IBM VSE/ESA 2.5 Performance Considerations'
'IBM VSE/ESA Performance on xSeries (NUMA-Q) Enabled for S/390'

The files are
VE13PERF.PDF, VE21PERF.PDF, VE21TDP.PDF, VEIOPERF.PDF, VEVMPERF.PDF, VEPERACT.PDF, VETCPPER.PDF, VESORTP.PDF, VECICSTS.PDF, VE25PERF.PDF, VEXEFSP.PDF

Notes ...

Disclaimer

This document has not been subjected to any formal review or testing procedures and has not been checked in all details for technical accuracy. Results must be individually evaluated for applicability to a particular installation.

Any performance data contained in this publication was obtained in a controlled environment based on the use of specific data and is presented only to illustrate techniques and procedures to assist to understand IBM products better.

The results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicability of this data in their specific environment.

The above disclaimer is required since not all dependencies can be described in this type of document.

Acknowledgements

Thanks to all who contributed directly or indirectly, be it by measurements, suggestions or in other ways.

Specific thanks to

Hanns-Joachim Uhl	doing all the performance runs
all CSI people	answering and clarifying all my questions

All mistakes and inaccuracies in this document are owned by me.

Please, as in the past, contact us if you have

- suggestions or questions regarding this document
- questions on VSE/ESA performance, not covered in any of the VSE/ESA performance documents

Wolfgang Kraemer, wkraemer@de.ibm.com
IBM VSE Development, Boeblingen Lab, Germany

Notes ...

Trademarks

The following terms included in this paper are trademarks of IBM:

ES/9000	ESA/390	System/390	AIX	Nway	
VM/ESA	VSE/ESA	ESCON	ECKD	CICS	...

Trademarks of other companies:

TCP/IP for VSE	Connectivity Systems Inc, Columbus Ohio (CSI)
UNIX	X/Open Company Limited
Windows	Microsoft Corporation
....	

References

The following are some references for further information in the context of TCP/IP.

General TCP/IP References

- TCP/IP Introduction, GC31-6080
- TCP/IP Tutorial and Technical Overview, Fifth Edition,
  by Eamon Murphy, Steve Hayes, Mathias Enders,
  Prentice Hall PTR, ISBN 0-13-460858-5, or
  ITSO Raleigh Redbook, GG24-3376-04, 06/95, 477 pages
- TCP/IP - Architecture, Protocols, and Implementation-
  by Dr. Sidnie Feit, 2nd Edition, McGraw Hill, ISBN 0-07-021389-5
- TCP/IP Illustrated, Volume 1 'The Protocols', by W. Richard Stevens, 576 pages. Addison-Wesley, ISBN 0-201-63346-9, 03/96
- Internetworking with TCP/IP,
  Vol I: Principles, Protocols, and Architecture,
  by Douglas E. Comer, 2nd Edition, Prentice Hall, SC31-6144-00
- IBM TCP/IP Performance Tuning Guide, SC31-7188-02, 03/97, 282 pages
  As TCP32PTG on the IBM MKTTOOLS tools disk
  (Addresses MVS, VM, AIX, OS/2, DOS, OS/400,
  Concepts, Tuning, Benchmark data)
- Using the Information Super Highway, ITSO Austin Redbook,
  GG24-2499-00, 05/95, 206 pages
- Introduction to TCP/IP, by Richard F.Lewis, IBM Washington.
  VM and VSE Tech Conf, 06/97, Mainz, Germany, Session 1B1,
  05/98, Reno, Nevada, Session 11F

References ...

References for TCP/IP with MVS

- IBM MVS TCP/IP -Performance Tuning Tips and Capacity Planning-,
  by L.Groner, D.Patel and R.Perrone from IBM Network Systems,
  SHARE 85, Session 2542, 08/95
  by L.Ferdinand, B.Kay, D.Patel, R.Perrone, S.Rimbey, IBM,
  SHARE 88, Session 3912, 03/97
  As TCPPERF PACKAGE on MKTTOOLS (for your IBM representative)
- IBM TCP/IP for MVS -Customization and Administration Guide-,
  V3 R2, SC31-7134-03, 09/96, 726 pages
- IBM TCP/IP V3.2 for MVS -Implementation Guide-,
  ITSO Raleigh Redbook, SG24-3687-03, 12/96, 661 pages

References for TCP/IP with VM

- IBM ICP/IP for VM -Planning and Customization-
  V2 R3, SC31-6082-02, 12/94, 352 pages
  V2 R4, SC31-6082-03, 12/96, 412 pages
- VM/ESA TCP/IP Performance, by Bill Bitner, IBM
  VM and VSE Tech Conf, 05/97, Kansas City, MO, Session 26E
  VM and VSE Tech Conf, 06/97, Mainz Germany, Session 37B
  SHARE 88, Session 9224, 03/97
  VM and VSE Tech Conf, 05/98, Reno Nevada, Session 26B
  http://www.vm.ibm.com/devpages/bitner/presentations/tcpip/
- VM/ESA V2.1.0 Performance Report, 12/95, p117
  TCP/IP file transfer via TCPNJE RSCS between 9121-320s using CTC with 3088
- VM/ESA V2.2.0 Performance Report, 12/96, p58
  TCP/IP 2.4 vs 2.3, File transfer from AIX to VM/ESA, with RPC UDP, using Token-Ring at a 3172-1
- TCP/IP for VM/ESA V2R3, by Alan Altmark
  (includes TCP/IP Level 310 enhancements)
  VM and VSE Tech Conf, 04/98, Frankfurt, Germany, Session 65E
  05/98, Reno, Nevada, Session 20A
- Getting Started with VM TCP/IP, by Alan Altmark
  VM and VSE Tech Conf, 05/98, Reno, Nevada, Session 20A

References ...

References for TCP/IP with VSE

IBM documents

- TCP/IP for VSE/ESA -User's Guide-, SC33-6601-00, 494 pages, 12/97.
  -01 available 07/98. Replaced by:
  TCP/IP for VSE/ESA -IBM Program Setup and Supplementary Information, SC33-6601-03
- The Native TCP/IP Solution for VSE,
  SG24-2041-00, ITSO Boeblingen Redbook, 223 pages, 08/97
  Obsolete, applies to CSI TCP/IP Rel 1.2.
  Replaced by the following redbook ..
- Getting Started with TCP/IP for VSE/ESA 1.4
  SG24-5626-00, ITSO Boeblingen Redbook, 235 pages, 05/2000
- VSE as a Webserver, SG24-2040-00, ITSO Boeblingen Redbook, 01/98
- Visit the WWW site:
  http://www.s390.ibm.com/products/vse/vsehtmls/tcphome.htm
  to check further information from IBM

CSI (Connectivity Systems) documents

- TCP/IP for VSE -Product Presentation-,
  WAVV Conference Cincinnatti/Ft. Mitchell, 10/99
  WAVV Conference Colorado Springs, 10/2000
- TCP/IP for VSE Manuals by CSI:
  Version 1.3 09/97, Version 1.4 11/99
  - Installation and Operation Guide
  - Commands
  - User's Guide
  - Programmers Reference
  - Messages and Codes
  - Optional Products Guide
- TCP/IP for VSE with Network File System. By Leo Langevin,
  Connectivity Systems Inc., 09/98
  VSE Customer Conference Call 09/16/98
- Visit the WWW site: http://www.tcpip4vse.com/
  to check further information from CSI

References ...

References for TCP/IP with VSE (cont'd)

Conference contributions

- TCP/IP Solutions for VSE/ESA
  by Boris H. Barth/Ingo Adlung, IBM Boeblingen.
  VM and VSE Tech Conf, 06/97, Mainz Germany, Session 52A
- TCP/IP for VSE and the Heterogeeous World of WERU AG
  IBM S/390 Enterprise Systems Bulletin, 03/97
- CGI Programming for VSE -Using VSE as a Web Server-
  by Leo Langevin, Connectivity Systems,
  VM and VSE Tech Conf, 05/97, Kansas City MO, Session 32B
- TCP/IP for VSE, Product Presentation, by Connectivity Systems,
  WAVV 97 Chattanooga, 11/97, WAVV 98 Albany, NY, 10/98
- WWW and VSE/ESA, by Anette Stolvoort, IBM.
  WAVV 97 Chattanooga, 11/97
- TCP/IP for 'Dummies', by Eric Vaughan, Intelliware,
  WAVV 97 Chattanooga, 11/97
- Web Enablement for VSE/ESA', by Eric Vaughan, Intelliware,
  WAVV 97 Chattanooga, 11/97
- NFS for VSE!, by Leo Langevin, Connectivity Systems,
  WAVV 97 Chattanooga, 11/97
  VM and VSE Tech Conf, 05/98, Reno, Nevada, Session 30H
- TCP/IP for VSE/ESA, Installation and Implementation,
  by Jon vonWolfersdorf, VM and VSE Tech Conf, 04/98, Frankfurt, Germany, Session 81J
- TCP/IP Socket programming with VSE/ESA, by Ingo Adlung.
  VM and VSE Tech Conf, 04/98, Frankfurt, Germany, Session 90K
  VM and VSE Tech Conf, 05/98, Reno, Nevada, Session 30B
- TCP/IP for VSE -The Last Word on Performance-,
  WAVV 99 Cincinnatti/Ft. Mitchell, 10/99
  WAVV 2000 Colorado Springs, 10/2000

Glossary

Glossary/Abbreviations

ACK	Acknowledgement
ARP	Address Resolution Protocol
CAF	CICS Access Facility
CGI	Common Gateway Interface
CETI	Continuously Executing Transfer Interface
CLAW	Common Link Access to Workstations
CTC(A)	Channel to Channel (Adapter)
FDDI	Fiber Distribution Data Interface
FTP	File Transfer Protocol
GPS	General Print Server
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LPR/LPD	Line Printer Requester/Daemon
LIBR	VSE Librarian
MAC	Medium Access Control
MIPS	Million Instructions per second, or Meaningless Indication of Processor Speed (if you misuse it)
MSS	Maximum Segment Size
MTU	Maximum Transmission/Transfer Unit
NFS	Network File System
OSA	Open Systems Adapter
OSI	Open Systems Interconnect

Glossary ...

Glossary/Abbreviations (cont'd)

PING	Packet Internet Groper
RPC	Remote Procedure Call
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TN3270	TELNET 3270
UDP	User Datagram Protocol
URL	Universal Resource Locator
WAVV	World Alliance of VM and VSE
XPCC	VSE Cross Partition Communication Control

New Documentation (TCP/IP 1.4) by CSI

All 1.4 documents are available on the CSI CD-ROM
and on the Internet via

http://www.s390.ibm.com/products/vse/support/tcpip/tcphome.htm

- TCP/IP for VSE 1.4 Installation Guide
- TCP/IP for VSE 1.4 User's Guide
- TCP/IP for VSE 1.4 Programmer's Reference
- TCP/IP for VSE 1.4 Optional Products (NFS, GPS)
- TCP/IP for VSE 1.4 Messages and Codes

TCP/IP General Intro -Performance View-

PART B.

TCP/IP General Intro
-Performance View-

Some Terms

Some Terms

Ù **Client**

„ **... a computer or process that initiates a request.**

Each client program makes requests to S/W running at a remote location

Ù **Server**

„ **... a computer or process that provides service to clients**

Ù **Daemon**

„ **... a program that 'listens' for requests from clients and then passes control to a server**

(A daemon is often called a 'server', since it is associated to a server, quasi controlling access, like in hell)

Ù **Host**

„ **... may be thought of as an end system (which gets a unique network address), not necessarily a mainframe**

E.g. - any VSE partition with TCP/IP
- any PC on a TCP/IP network

Protocol Layers for Internet

Protocol Layers

The following likewise applies to any TCP/IP network

```
        Layer
    ----------------                 ------------------------------
   |                |               |                              |
   |                |               | Gopher                       |
   |                |               | TELNET            SNMP       |
   |                |               | FTP(TCP)          FTP(UDP)   |
   |  Application   |    MESSAGE -> | HTTP(WWW)         NFS(RPC)   |
   |                |    ........   | SMTP(E-mail)      ...        |
   |                |               | ...             |            |
    ---------------                  ------------------------------
   |                | UDP DATAGRAM->|                 |            |
   |                | TCP SEGMENT-> |                 |            |
   |  Transport     |    ........   |    TCP          |  UDP       |
   |                |               |(connection      |(connection-|
   |                |               |       oriented) |      less) |
    ----------------                 ------------------------------
   |                |               |                              |
   |  Internetwork  | IP DATAGRAM-> |             IP               |
   |                |    ........   |      (connectionless)        |
   |                |               |                              |
    ---------------                  ------------------------------
   |  Link          |               |                              |
   | (Device Driver)|               | Token-Ring, Ethernet, FDDI,  |
   |                |    FRAME ->   | ATM                          |
    ................     ........    ..............................
   .                .               .                              .
   .  Hardware      .               .                              .
    ----------------                 ------------------------------
```

This layer model here is same as the OSI model,
except that OSI shows Presentation and Session as separate layers
(here included in Application layer).

Protocol Layers for Internet ...

Protocol Layers

Both TCP and UDP are on the 'Transport Layer'.

Ù TCP = Transmission Control Protocol

Accepts data transmission requests of any length

Breaks the transmission data into chunks (TCP segments)

Reliably sends them across the network

Employs checksums, sequence numbers, timestamps, timeout counters for retransmission

Uses and exploits ACKnowledgements for 'windowing'

ACKs used are always
- cumulative, i.e. not selective
- positive, i.e. no negative ACKs

(Many TCPs send ACKs for every 2nd data segment it receives)

Í Connection oriented

Protocol Layers for Internet ...

Protocol Layers (cont'd)

Ù UDP = User Datagram Protocol

UDP datagrams treated as 'single entities'

Each UDP datagram directed separately to the receiving application

No checking for successful delivery, no usage of ACKs

UDP provides Send space and Receive space. If space full, extra data is discarded

- Inbound:

  Data moved from 'Receive Space' in UDP layer (Receive Buffer) to User Data Buffer in application

- Outbound:

  Sender does not know when receiving buffer is full. Receiver discards extra incoming data, to be retransmitted

Í Connectionless

Less frequently used,
but used e.g. for NFS in TCP/IP for VSE/ESA

Less reliable, but potentially faster

Protocol Layers for Internet ...

Ù IP = Internet Protocol (Network Layer)

„ Creates a virtual network view

„ Has no reliability, flow control or error recovery,

i.e. no timeout, no retransmission

„ Can do fragmentation and reassembly of its datagrams

Loss of a fragment causes ALL fragments to be re-transmitted
(no ACK mechanism provided on fragmented IP datagram level)

- IP transmission protocol requires that each 'data packet' either be delivered in a timely fashion or thrown away

Í Just performs the transfer of IP datagrams

Ù Encapsulation principle for layers:

Each layer

„ sends its data down the protocol stack by adding header info to the data ('outbound')

„ receives its data from the layer below by looking at certain identifiers and by removing its own headers ('inbound')

Frames, Datagrams, Segments

„ **Physically Transferred (Frame)**

```
| Phys.   |                                       |      |
| network |  IP datagram (or fragment) as data    | trai-|
|   hdr   |                                       | ler  |
          |<--------- Maximum Transfer Unit (MTU) -----/-->|
```

„ **IP Datagram**

Often also called 'Packet'

```
| IP hdr    |              IP data                  |
| 20 byte   | (TCP segment or UDP datagram or ...)  |
|(or more)  | (potentially to be fragmented by IP)  |
```

```
IP header:
      - IP protocol version               - header length (<60)
      - type of service (priority, ...)   - fragmentation info
      - type of higher level protocol     - header checksum
```

- IP does not impose a maximum IP datagram length, but all subnetworks must be able to handle at least 576 bytes

„ **TCP Segment**

```
| TCP hdr   |                         |
| (20 byte) |        TCP data         |
            |<-- Max. Segm. Size (MSS) -->|
```

```
TCP header:
      - source/destination info          - sequence number
      - header size                      - checksum
```

- From the receiving TCP an ACK is required for each segment
   -> A byte data stream is composed of multiple TCP segments

„ **UDP Datagram**

```
| UDP hdr   |                         |
| (8 byte)  |        UDP data         |
```

```
UDP header:
      - source/destination info           - length ....
```

- Composition of data from different UDP datagrams and control of transmission is NOT part of UDP, to be done by application

Frames, Datagrams, Segments ...

MTU (Maximum Transfer Unit)

„ **Maximum amount of data in a frame that can be sent over the physical media**

and thus ...

Max. IP datagram size

(w/o fragmentation by local IP)

Adapter Type	Default	Minimum	Maximum (examples)
Ethernet	1500	576	1500
Token-Ring	1500	576	about 4000 (4 Mbit/sec) about 8000 (16Mbit/sec)
FDDI	1500	576	2000
CTC	4096	576	16K (RS/6000 CLAW) 32K (S/390 CTCA)
OSA-2	1500	576	

- All sizes in byte
- For any P/390 or R/390 simulated 3172, MTU must not exceed 1492

If a connection is across multiple nodes ...
the smallest MTU of any data link that the connection uses will be relevant: 'Path MTU'

MSS (Maximum Segment Size)

„ **Biggest amount of data a TCP stack can receive in a single TCP segment**

This value is sent at session setup to the TCP partner, who has to observe this value (Default is 536).
Assuming for the moment a constant IP hdr size of 20 bytes:

```
|<-- Biggest non-fragmented IP datagram = MTU --->|
| IP hdr   | TCP hdr  |            MSS            |
| 20 byte  | 20 byte  |                           |
           |<---------- TCP Segment ------------->|
```

MSS = MTU - 40 bytes (if w/o fragmentation by local IP)

Frames, Datagrams, Segments ...

MSS (cont'd)

The max. amount of data TCP can put into a single TCP segment (w/o requiring later IP fragmentation) is in general

MSS + 40 - (size of TCP + IP headers)

Resulting MSS Value for a TCP Connection

„ **Scenario**

When establishing a TCP connection, the server and client exchange info that specifies the maximum 'packet' size each can receive:

the MSS value from the other system.

This value is considered when a partner TCP/IP sends TCP segments out.

„ **Optimal MSS for a TCP Connection (Direction)**

```
               - the MSS value of the other system
MIN out of
               - the MTU-value of the route minus 40 byte
```

The resulting actual maximum size of TCP data is usually optimal, if ...

- it is as large as possible, but without requiring any IP fragmentation (and thus no reassembly) along the path from source to destination

IP Fragmentation and Reassembly

IP Fragmentation and Reassembly

Example (MTU=620)

```
                  Large (unfragmented) IP-Datagram
| IP      |                data (900 byte)             |
| header  |          data 1 (600)  |  data 2 (300)     |
                              |
                              V
| IP fragm. |  data fragment 1  |
|  header   |   (600 byte)      |
  IP header has the 'more fragments bit' set, plus offset 0
                              +
                              | IP fragm. |data fragment 2|
                              |  header   |  (300 byte)   |
                              IP header has set offset 600
```

- Large IP datagrams ('packets') can be fragmented, each getting its own header
- Transmissions via gateways through other networks should use the 'default TCP/IP packet size' of 576, unless all intervening gateways and networks are known to accept larger packets
- The complete datagram is restored
   - only at the final destination (reassembly)
   - as soon as all fragments have arrived at the IP level

Performace Impacts

For sender

- CPU overhead to create and transmit additional packets
- Retransmit ALL packets in a datagram if a packet is lost

For receiver

- CPU overhead to re-assemble the packets
- Memory overhead for buffers to re-assemble the packets
- Delays if a packet is lost

If fragmentation only occurs occasionally, no problem

TCP/IP Window Technique

TCP/IP Window Technique

Ù **Send as much data as possible/reasonable before waiting for an ACKnowledgement**

In case of TCP, this principle is applied on the TCP level.
(Thus, here we talk of segments instead of packets).

Largest TCP window size is 64K, except 'Window Scale' would be used (RFC 1323).

Ù **General**

- A Receiver decides how much data it is willing to accept
- A Sender must stay within this limit
- A Window is always related to a single session and direction
- At connection setup, each partner assigns receive buffer space
  (usually a multiple of the maximum segment size)
- Every ACK sent back by the receiver
  - contains the highest segment number received 'in sequence'
  - and the size of its current receive window left

Sliding and Breathing Window (SEND)

```
                               -----------------
                              |                 | -->
         TCP      ----------------------------------------------
         segment  |  1  |  2  |  3  |  4  |  5  |  6  |  7  |
                  ----------------------------------------------
                              |                 | -->
                               -----------------

 Sent          <-------------------->
 ACK received  <------->
 Can be sent immediately             <---->
 Send when window moved                     <-------------
```

Cont'd on next page

TCP/IP Window Technique ...

TCP/IP Window Technique (cont'd)

Sliding and Breathing Window (RECEIVE)

```
                            -----------------
                           |                 | -->
      TCP      -------------------------------------------------
      segment  |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |
               -------------------------------------------------
                           |                 | -->
                            -----------------

 Received      <---------------------->
 ACK sent out  <---------->
 Can be received immediately            <--->
 Can be received after the                    <-----------------
  RECEIVE window moved
```

Ù **TCP Windowing Rules (SEND)**

- Send out all segments within the current window, independent of any ACK
- For each segment sent, start a time-out timer.
  Retransmit segment after time-out, if no ACK received
- Move/Adapt current window based on
  - highest ACK received
  - changed window size (if so) in last ACK
- Sizes/Number of Send and Receive buffers on TCP layer determine maximum window sizes
- Maximum window sizes also depend on platform

PS/2	16384 (fixed size)
RS/6000	4096, 16384, 32768

Í **Effect similar to Pacing in SNA-networks**

TCP/IP Protocol Layers (Revisited)

TCP/IP Protocol Layers (Revisited)

Summary and Location of Activities

```
                   ----------------------------------------
                  |     TCP APPL.           UDP APPL.      |
                  |                       |                |
 Application      |                         (ACKs)         |
 Layer            |                       | (Sequenc.)     |
                  |  out           in       (Sess.ctrl)    |
                  |   |             A     |                |
                  |   |             |        |      A      |
                   ---|-------------|--------|------|------
                  |   | Byte stream |     |  |      |      |
                  |   V             |        V      |      |
                  |                       |                |
                  |   Session control                      |
  Transport       |    Sequencing         |                |
  Layer       TCP |     Checksum             Checksum (opt)| UDP
                  |    Windowing+ACKs     |                |
                  |Retransmission                          |
                  |                       |                |
                  |  Add..       Remove..   Add..   Remove.|
                  |       TCP hdr         |    UDP hdr     |
                  |   |             A     |  |       A     |
                  |   |             |'MSS'|  |       |     |
                   ---|-------------|--------|-------|-----
                  |   |   TCP       |        | UDP   |     |
                  |   V   segments  |        V data- |     |
                  |                            grams       |
                  |                                        |
                  |                                        |
  Internetwork    |    Fragmentation     Reassembly        |
  Layer           |         'Next hop routing'             |
              IP  |         IP header checksum             | IP
                  |             <--------                  |
                  |    Add..             Remove..          |
                  |          |   IP hdr    A               |
                  |          |'MTU'        |               |
                   ----------|-------------|---------------
                  |          |   IP        |               |
  Link            |          V   datagrams |               |
  Layer      Link |                                        | Link
                  |      H/W internal: Ethernet   hdr      |
                  |                    Token-Ring hdr      |
                   ----------------------------------------
```

```
 Outbound fragmentation may occur if
                        MTU-40byte < MSS of other system
 Inbound reassembly may occur if
                        MTU-40byte < MSS of VSE host
```

TCP/IP Concepts and Algorithms

Basic Concepts of TCP/IP

On order to better understand potential effects of parameter selection, it is very helpful if some basic TCP/IP concepts are understood.

These concepts include

- Frames, Datagrams and Segments

- Fragmentation and Reassembly

- Send and Receive Buffer management via Window sizes and Acknowledgements

These concepts have been schematically sketched in the previous charts.

Performance Algorithms for Communication

Several of the many performance algorithms for TCP are discussed in Dr. Sidnie Feit's book, pp 233 to 243.

They include

- Delayed ACKs
- Duplicate ACKs
- Slow Start
- Silly Window Syndrome
- Nagle Algorithm
- Retransmission Timeout
- Exponential Backoff

Regarding the ACKs, refer to a following chart.
The other algorithms are advanced and are not discussed here.

For faster links with very high transmission rates, there exists a Request For Comment (RFC 1323)

'TCP Extensions for High Performance'

which includes

- Window Size Limit (using an implicit scale factor)
- Selective (vs cumulative) ACKs

TCP/IP Performance Tuning

TCP/IP Performance Tuning

TCP/IP performance is influenced by a number of parameters that can be tailored for the specific operating environment. In general these tuning parameters can be grouped into ...

„ **Operating System tuning**

Operating System tuning should be familiar to most VSE/ESA customers, thus needs not to be mentioned in more detail here.

It also may include to tune local VSE file attributes, especially for the purpose of TCP/IP file transfers.

„ **TCP/IP setup tuning**

TCP/IP tuning is expected to be a new experience for many VSE customers.

It refers mostly to the optimal setup of a TCP/IP partition.

„ **Communication/Network tuning**

Mainframe end
Network
Workstation end

Communication (or Configuration) tuning is closely related to TCP/IP setup tuning.

It refers to the configuration (including links etc) of the network, and may be also the parameter selection on the other side, where also a TCP/IP resides.

„ **TCP/IP Application tuning**

This is only possible, if the customer can influence the TCP/IP application, e.g. at the Sockets level.

Practical Performance Aspects

Practical Performance Aspects

Ù **TCP/IP performance is limited by the**

„ **speed of the slowest link**

Throughput potentially is greatest when using FDDI for LANs.
Ethernet and 16Mbps Token-Ring networks are about comparable, lowest throughput usually is obtained for 4 Mbps Token-Ring

„ **window size of the receiver, divided by the round trip time**

„ **amount of CPU-time available for TCP/IP on host**

„ **speed of reading/writing data from/to disk**
(e.g. FTP)

Ù **Note**

Many TCP/IP performance problems are

- environment specific
- implementation deficiencies

- not caused by inherent protocol limits

(Partridge and Pink 1993)

Network Performance

Network Performance

Long transfer times in a net may be caused by ...

(set aside too heavy network traffic)

„ **Slow links or small MTU sizes**

„ **Too many links involved/broken or Routing not efficient**

„ **Inefficient setup of packet and window sizes**

„ **Higher share of discarded IP datagrams**

E.g. since 'time_to_live' expired

„ **Higher share of resent TCP segments**
'Retransmission rate'

E.g. ACKs are delayed too long

TCP/IP Acknowledgement Consideration

TCP/IP Acknowledgement Consideration

Here some re-visited and more info on TCP ACKs.

Ù **Background Info (Re-visited)**

„ **TCP ACKs are 'cumulative' (not selective)**
The receiver only tells the number of the 'lowest' missing packet (=TCP segment).
No indication included on any 'higher' packet received.

„ **No packet must be individually and immediately ACKnowledged**
The RFC says that the delay of an ACK should not be more than 500 ms

„ **Packets are only sent, as long as the receiver's window can hold the data**

„ **Packets are re-sent, if after a timeout no ACK was received by the sender**
Sometimes earlier, when multiple ACKs refer to the same segment

Ù **Performance Implications**

„ **Sender should proceed to send data, as long as receive window is open and not too huge**

„ **A too low timeout in the sender may cause unnecessary re-transmissions of packets**

„ **A too high timeout may reduce the data rate**
- especially when transfer is unreliable
- when receiving window is small or not enough data is sent before the sender waits for an ACK

TCP/IP Principal Perf. Dependencies

TCP/IP Principal Performance Dependencies

The following is a list of principal parameters which tries to globally categorize performance/tuning impacts.

```
Overall Performance
                     Response Times, Elapsed Times,
                     Throughput,
                     Resource Consumption
```

is determined by the components shown:

Parameter (type)	Host CPU-time	Host Storage	Network Transfer time	DASD time *
Host CPU speed S/390 Op.Syst. & setup	X X	- X	- -	- x
MTU/MSS used Window size #Xfer buffers	X	x x X	X X x	- - -
Type of Comm. Adapter Network/Line speed Network reliability #Appl.-bytes in/out	 X X	 x X	X X X X	- - - X
TCP/IP implementation TCP/IP application Other TCP/IP parameters	X X X	X X X	X X X	X X X
DASD I/O Subsystem DASD I/O Blocking	- x	- -	- -	X X

```
X  means major impact
x  means smaller or secondary impact
-  means no or negligible impact     ... in general

- Transfer time here includes wait for transfer

* DASD time only applicable if DASD involved (e.g. FTP)
```

Overall Capacity is also of interest and of specific importance for multiple concurrent sessions (e.g. TN3270)

TCP/IP Communication Tuning

TCP/IP Communication Tuning

Improving communication performance for a given network:
- increasing Throughput
- improving Response Times

High Impact of
- Send and Receive Buffer/Window Sizes
- Packet Sizes

Ù Optimum/Minimum Window Size

Round-trip times

Round-trip times for IP datagrams (obtained e.g. via PINGs, refer to separate chart) roughly correspond to the average time between the sending of an IP datagram and receiving the ACK, sent from the partner TCP and the originating TCP, even better, when average datagram sizes are used for PINGs.

Calculation

Roundtrip times can be used to roughly determine optimum (or better minimum) buffer sizes:

If you could transfer (instantaneously) P MB per sec via TCP/IP through the network (link(s)), and you need Tping msec to transfer each IP datagram, then, roughly

```
P MByte/sec  x Tping msec  = P x Tping  KByte
```

should be at least the window size (for SEND, and for RECEIVE); also-called 'Bandwidth*Delay product'.

(This simply is the average maximum amount of data which may be sent w/o acknowledgement)

Example

```
Assume   - a (slowest) network link of 8 Mbit/sec = 1.0 MB/sec
         - an approximate PING time of 20 msec

              1.0 MB/sec x 20 msec = 1.0 x 20 KB = 20 KB
```

In this case, about 20 KB would be required as window size

TCP/IP Communication Tuning ...

TCP/IP Communication Tuning (cont'd)

Ù Optimum/Minimum Packet Size

Avoid that packets usually are fragmented. Small fragmentation still may be acceptable.

```
- FTP, NFS:   Bigger packet sizes are desired.
- TN3270:     Increasing the allowed packet size may not help
              and potentially wastes virtual storage.
```

Refer to the previous considerations.

Ù 'The bigger the better'?

When starting with small buffer sizes and/or packet sizes, this usually holds.

But this is only true as long as

```
     - the network is relatively reliable
 and
     - no fragmentation is forced
 and
     - there is enough real storage to back up the increased
       total buffer sizes
 and
     - you do not have reached a performance limit, determined
       by any other resource
```

TCP vs UDP Performance

„ UDP may (formally) have less overhead

The overhead for managing a connection-driven environment is not included in UDP itself.

„ Note of care

Some application (must) provide connection management, so this CPU time is just required on the application layer.

Send and Receive Buffers (MVS and VM)

Send and Receive Buffers (MVS and VM)

Terminology and parameters from the TCP/IP for MVS and VM products

Type	Parameter (MVS and VM terms)	Number Deflt. Rec.	Size	Purpose
Data Buffers	DATABUFFERPOOLSIZE	n 160 -	8..256K 16K 32K	Regular data
Small Data Buffers	SMALLDATABUFFERPOOLSIZE	n 0 -	2048 - -	*S)
Tiny Data Buffers	TINYDATABUFFERPOOLSIZE	n 0 -	256 - -	*T)
Envelopes	ENVELOPEPOOLSIZE	n 750 -	2048 - -	*E)
Large Envelopes	LARGEENVELOPEPOOLSIZE = MTU	n 50 -	8..64K 8K -	*L)

```
*S) Used for TELNET and Offload function, overflow to regular
*T) Used for Offload function
*E) Used for UDP datagrams >2K
*L) Used if packet (UDP datagram) does not fit into Envelopes

- Data are discarded, if 'Data' or 'Large Envelopes Buffers'
  are exhausted
- The size of the Large Envelopes also determines the MTU size
```

IBM 3172 Specifics

3172 Modes of Operation with TCP/IP

Ù **ICP Mode**

The software in the 3172 is the IBM Interconnect Controller Program (ICP)

- Short data blocks received are packed into frames of up to 20K before sending them over the channel to the host. Before the next frame is sent, a DE (device end) has to be waited for (in contrast to CLAW or CETI).

  Maximum response length: Frames smaller than that are sent directly to the host w/o delay. Optimal: 500 byte. Default: 100 byte

  Block delay time: Amount of delay which is allowed while received frames are blocked for retransmission. Optimal: 10 msec. Default: 20 msec

- Configure the adapters to reject traffic not explicitly addressed to it. This will avoid unnecessary CPU-time overhead.

Ù **Offload Mode**

Software is OS/2 with the Offload Feature for TCP/IP for MVS or VM

- Moves some TCP/IP processing from MVS or VM to the 3172-3. Some S/390 CPU-time reductions (from SC31-7188):
  12 - 15% for MVS and VM using Telnet
  30 - 50% for MVS and VM using FTP
  Note that using 3172 Offload may show reduced throughput, up to 30% for FTP

Ù **OSA-2**

- An integrated H/W feature
  - Looks to S/W as 3172, does not have the Offload function.
  - Avoids inspection of IP traffic to other hosts (filtering)

TCP/IP for VSE/ESA Product (TCP4VSE)

PART C.

TCP/IP for VSE/ESA Product (TCP4VSE)

Here, the TCP/IP for VSE/ESA product from Connectivity Systems is referred to:

'IBM TCP/IP for VSE/ESA Vers.1 Release 3' (Pgm number 5686-A04).

It is key-enabled and part of the VSE/ESA 2.3/2.4 base and available as

- Base Pak
- Application Pak (includes Base Pak functions)

Available since 2000-06-16, via PTF UQ44071, is a major enhancement

'IBM TCP/IP for VSE/ESA Vers.1 Release 4' (Pgm number 5686-A04).

Í **New connectivity capabilities for VSE/ESA**

Highlights

Highlights

Ù **VSE native implementation**

Ù **Especially developed for VSE**

Ù **Runs in a separate VSE partition**

„ **Own multitask mechanism**
Uses several VSE subtasks (plus internal 'pseudo-tasks')

„ **All daemons/servers run in the TCP/IP partition**

„ **Each TCP/IP partition has a unique ID in the EXEC card**
Links between TCP/IP partitions can be configured

„ **XPCC is used to communicate with POWER**
Other partition communications are done via XMOVE

Ù **More info**
Refer e.g. to the official TCP/IP for VSE literature, or

- TCP/IP for VSE/ESA -User's Guide-, SC33-6601-00, 12/97, -01 available 07/98
- TCP/IP Solutions for VSE/ESA
  by Boris H. Barth, ITSO Boeblingen
  VM and VSE Tech Conf, 06/97, Mainz Germany, Session 52A
- IBM S/390 Open Systems Adapter -Rerformance Report-.
  As OSAPERF PACKAGE on MKTTOOLS, available to your IBM representative, 11/96

Supported Environments

Supported Environments

Ù **VSE/ESA 2.3 and up (IBM shipped version)**

VSE/SP and VSE/ESA releases (CSI V1.3 shipped version)
VSE/ESA releases 1.3 and up (CSI V1.4 shipped version)

- CSI TCP/IP for VSE versions running on VSE/SP have no Librarian API (other functions are supported)
- VSE/SP is no more supported, and has major storage restrictions (24 bit, plus VTAM in shared space).
- S/370-mode is formally available only in VSE/ESA 1.4

Ù **Communication H/W**

„ **3172/8232 LAN Channel Station Controller**

- Token-Ring, FDDI, Ethernet
- 3172 emulation by PC Server S/390 systems (P/390, R/390)

„ **ES/9221 Integrated Adapter (CETI)**

Token-Ring, X.25, Ethernet

„ **OSA-2 (Oct 95)**

- Token-Ring (4/16 mbps), Ethernet(10/100 mbps). FDDI and ATM (LAN Emulation only).
- OSA/SF is highly recommended, partly required
- OSA-Express (since 06/99) NOT supported by VSE/ESA

„ **2216 Nways Multiaccess Connector**

Supported Environments ...

Ù **Communication H/W (cont'd)**

Channel attachments to S/390 are fast and appear as 2 adjacent devices (for input and output).
(They also include 2 buffer areas for concurrent transfers):

„ **CTCA to any S/390 operating system**

Maybe even a virtual CTCA if both under same VM.

```
Note: No need to care for the VSE MIH setting
      (MIH is always disabled for CTCs).
```

„ **Channel attached RS/6000 (CLAW)**

TCP/IP Application Types

TCP/IP Application Types ('Internet Services')

Ù **TELNET (Client and Server)**
Terminal access from and to VSE systems

Ù **FTP (Client and Server)**
Transfer files from and to VSE systems

Ù **GPS (Server)**
Direct VTAM printer data to any TCP/IP printer

Ù **Intranet/Internet Server**
Access from TCP/IP network to HTML objects/data under VSE

„ **HTTP Server**

- VSE as Web server in Internet

Ù **LPR/LPD (Client/Server)**
Print on any TCP/IP printer / on a remote VSE system
via Line Printer Requestor/Daemon

Ù **NFS (Server only)**
Access data stored in VSE as if it were local.
Appears to DOS, Windows etc as a drive, to UNIX as a subdirectory

All these applications use the TCP protocol, except most of NFS, which uses UDP.

Ù **APIs (Sockets)**
For major programming languages, building TCP/IP applications

TCP/IP for VSE Partition

TCP/IP for VSE Partition

```
        --------------------------------------------
       |        TCP/IP for VSE partition             |
       |        - - - - - - - - - - - - -            |
       |        ID=xx (unique, default=00)           |
       |                                             |
       |                                             |
       | TCP/IP Code/Control Blocks/Areas/Buffers    |
       |                                             |
       | Daemons (Server):                           |
       | - - - -                                     |
 <===>|  TELNET Daemon(s) 1/conc. session            |
       |                                             |
 <===>|  FTP Daemon(s)    1/conc. session            |
       |                    (in or out)              |
 <===>|  GPS Daemon(s)    1/VTAM printer             |
       |                                             |
 <===>|  HTTP Daemon      1/port used                |
       |                                             |
 <===>|  Gopher Daemon    1                          |
       |                                             |
 <===>|  LPD Daemon(s)    1/virtual printer          |
       |                                             |
 <===>|  NFS Daemon       1                          |
       |                                             |
       |                                             |
       | Client Mgrs:                                |
       | - - - - - -                                 |
 <===>|  TELNET Client Mgr                           |
       |                                             |
 <===>|  FTP    Client Mgr                           |
       |                                             |
 <===>|  Gen.Purpose Client Mgr (LPR,...)            |
       |                                             |
       |                                             |
        --------------------------------------------
```

1 TCP4VSE daemon/server can have -at one point in time- ...

```
   - only a session with 1 client:
           TELNET, FTP

   - sessions with multiple clients:
           HTTP, LPD, NFS
```

TELNET

TELNET

- Teletypewriter Network, does not support graphics.
- Makes the user's terminal (Client) appear as a local terminal
- TN3270 is TELNET with 3270 emulation:
  pass 3270 screen data and keyboard inputs

Ù **As Server (Daemon)**

„ **Allow remote access/logon from any TCP/IP to VTAM applications via TN3270**

• Runs as 'subtask' in the TCP/IP partition

• 1 concurrent TN3270 session requires
  - 1 TELNET daemon, defined via DEFINE TELNETD
    (only 1 session per socket is inherent to TCP)
  - 1 VTAM APPL-id
  - 1 VTAM terminal LU-name

• VTAM 4.2 needs 1MB of addt'l dataspace for each TCP/IP partition running TELNET daemons (runs as a VTAM appl.)

• CICS Access Facility (CAF, not part of VSE/ESA 2.3.0 GA)

  - Would allow TN3270 to bypass VTAM
    (TCP/IP appears to CICS as a TOR).

Ù **As Client**

„ **Access to other applications (on 3270 or UNIX platform) from local CICS**

• VSE users (in CICS, also batch) get

  - full 327x emulation (connecting to VM/MVS and VSE) if a 3270 session negotiation with the foreign host is successful

  - Network Virtual Terminal (TTY line-)support otherwise

General Print Server (GPS)

General Print Server (GPS)

Ù **Allows, in TN3270 environments, to direct VTAM 328x print data to any TCP/IP capable printer**

- Function not available/possible within TN3270 daemons
- No change in application req'd

Ù **Method of Operation**

- A GPS daemon in TCP/IP partition

```
  - identifies itself to VTAM as a locally attached 3287
    printer
  - intercepts/gets VTAM non-SNA print data and reformats it
  - uses the LPR/LPD protocol
```

Ù **Function available with APAR PQ27233 (99-07-16).**

A TCP/IP for VSE/ESA feature

```
- Key protected
- Priced
```

Ù **Performance**

Service Pack L allows to select, where the print data is stored, before it is sent out:

```
   QUEUING=MEMORY|DISK  In TCP/IP GETVIS-31 | VSE library
```

Regarding GPS virtual storage requirements, refer to separate discussion.

FTP

FTP

Transfer data or files from/to remote systems

Requestor is client.
File transfer (uses TCP) can be any direction (GET/PUT).

Ù **As Server (Daemon)**

„ **Allow bi-directional access from FTP clients to data/files under local VSE**

1 FTP daemon required for each concurrent FTP session, defined via the DEFINE FTPD command.
A long running task, listening to FTP requests from any client, even when initiated locally.

Ù **As Client**

„ **Starting an FTP session (initiate a file transfer) between VSE and a remote system**

```
via  - FTP client outside of VSE
     - CICS txn  (-> Interactive FTP client)
     - Batch job (// EXEC FTP or FTPBATCH)
                 (-> Internal or External Batch FTP client)
     - Program APIs
     - A defined event (POWER LST or PUN)
       (-> 'Automatic FTP', Service Pack H, 08/98):
       e.g. DEFINE EVENT,ID=AFTP,TYPE=POWER,CLASS=F,-
            QUEUE=LST,ACTION=FTP
       Refer to APAR II11362 for a detailed description.

-> Interactive, automatic, programmatic, or via Batch
```

„ **Sequence of user operations**

```
Here an example for remotely initiated FTP:
     - Connect to a remote host
     - Select a directory
     - List files available for transfer
     - Define the transfer mode
     - Copy files from/to the remote host (GET/PUT)
     - Disconnect from the remote host
```

FTP ...

FTP (cont'd)

Ù **Supported file types**

FTP itself does not know file characteristics, only the server:

```
       - VSAM ESDS and KSDS
       - VSE SD files
       - VSE libraries (via LIBRM I/F)
       - POWER files (job submission or retrieval of listings)
       - VSE/ICCF libraries (Read only)
```

TCP/IP for VSE does not support the very simple level of functionality of TFTP (Trivial FTP) using UDP datagrams.

Ù **FTP Setup Comparison**

	Type of Files	File I/O (FTP Daemon) in	FTP initialized in
Interactive FTP	DEFINE FILEd files	TCP/IP part.	FTP client *1
Batch FTP (// EXEC FTP)	- " - or 'Autonomous'	- " -	Batch part.
'FTPBATCH' (// EXEC FTPBATCH)	'Autonomous' or (ServPack L) DEFINE FILEd	Batch part.	Batch part.

```
  - 'Autonomous' is specification via locally defined DLBLs
    (but, consider security aspects)
  - TCP and IP activity is always in the TCP/IP partition
 *1 FTP client is outside VSE or a CICS txn
  - For FTPBATCH (ServPack J etc) refer to Info APAR II11596
  - Automatic FTP fully runs in TCP/IP partition
```

Í **FTPBATCH allows**

- best load balancing vs other TCP work

Especially when only 1 TCP/IP partition

- highest aggregate data rates for multiple FTPs

Due to multiple File-I/O routines

HTTP (Web Server) and Gopher

HTTP

Ù **As Server (Daemon), only**

„ **VSE as Web server in an Inter/Intranet i.e. storing HTTP objects in VSE libraries**

```
- HTML documents
- JPEG/GIF/TIF
- JAVA or other objects
- Video etc.
```

1 HTTP daemon required regardless of the # of Web sessions, defined via the DEFINE HTTPD command.

It is expected that most of the HTTP objects will reside in a VSE Library.

Requests to the Web server are issued from Web browsers outside the VSE host:

Web server and client communicate using HTML

Gopher

Ù **As Server (Daemon), only**

„ **Access from remote systems to data/files under local VSE**

The Gopher client uses easy-to-use menus, and both are using the Gopher protocol.

1 Gopher daemon required for any number of Gopher sessions, defined via the DEFINE GOPHERD command

LPR/LPD

LPR/LPD

Ù LPD (=Server or Daemon)

„ Print data of any TCP/IP system on a VSE printer

1 LPD daemon required for each virtual printer, defined via the DEFINE LPD command.

LPD interfaces

- with the POWER LST queue (printing is controlled by POWER)
- with a disk-based VSE file: 'print' to a file

Ù LPR (=Client)

„ Print VSE data on any TCP/IP network printer

Invoked

```
  - automatically   AUTOLPR, monitoring POWER LST classes
                    (done via a generic GET for a class,
                     every 45 sec).
                    Service Pack K allows to modify this
                    interval via SET AUTO_TIME = nnnn.

  - via CICS txn    'LPR'-txn

  - via batch job:  // EXEC CLIENT,PARM='ID=0x,APPL=LPR'
                    lpr command
                          ...
```

NFS Server

NFS

In TCP/IP for VSE/ESA, Serv.Pack G, 07/98, separate product key

As Server (Daemon), only

„ Transparent access from NFS client (PC or UNIX) to files stored in a remote VSE as if it were local:

'Share file systems across a TCP/IP network'

„ NFS assumes a hierarchical file system, with each file being a byte stream of certain length, essentially w/o record structure

File names and structures are automatically converted to what is normal to the client.

NFS itself is NOT an Data Base Access method, just an access method for total files.
(Single records of a VSE file only theoretically could be retrieved, but only if the byte offset in a file and the exact length would be known to the PC or UNIX application).
Depending on the interfaces used, also VSE members within a single VSE file can be accessed.

This lack of record positions in a file causes that upon record changes in a file, usually the entire file is being written.

In PC and UNIX land, logical records are delimited by indicators.
They may be added by NFS for VSE at the end of each record.

„ NFS Implementation

```
1 NFS daemon required in total, defined via DEFINE NFSD.
It uses:
           - Remote Procedure Call (RPC)
           - The NFS V2 protocol
           - The UDP transport protocol
             (Sequence of packets ensured by NFS,
             TCP used only to setup communication)
           - 31-bit GETVIS storage

to access LIBR and POWER members, and VSAM ESDS files
```

NFS Server ...

NFS (cont'd)

„ Scenario

RPC API allows to call subroutines that are executed on a remote system. A caller (client) sends a call message to the server process and waits for a reply message.

The NFS client first initiates the MOUNT protocol

```
    - to 'mount' any remote item, e.g.
                 - a VM minidisk
                 - a VSE library/sublibary
                 - a VSE/VSAM file

      as a new local subdirectory (UNIX)
      as a new drive's root (DOS, Windows, OS/2)
```

and then the NFS protocol

```
    - to actually do basic I/O operations to a remote file
      e.g. LOOKUP search
           READ and WRITE
           RENAME, REMOVE  ...

You may e.g. - edit VSE library members with Notepad
             - look at POWER lsitings in Word
             - use VSAM files in EXCEL
```

„ READ/WRITE

NFS client itself has no idea whoelse is updating records in a (source) file which was mounted in NFS.
Thus READs may be 'dirty READs'. Use of a VSAM file (here ESDS) with appropriate SHROPT definition would avoid that.

The VSE NFS server will do synchronous ('immediate') WRITEs to ensure file integrity.

„ NFS Server vs LANRES Virtual PC Disk

- LANRES data are logically not understandable by VSE (i.e. is a separate 'subset' of files)
- NFS access is concurrent to VSE native access (i.e. data can also be used by NFS clients)

Socket APIs

Socket APIs

Ù SOCKET macro

Essentially, the SOCKET macro acts as a program API to communicate with clients in the TCP/IP partition, mostly from an application running outside the TCP/IP partition ('external sockets').

Assembler, COBOL, PL/I and C programming languages can be used.

Naturally, the de facto standard 'BSD Sockets' is supported by TCP/IP for VSE/ESA.

SOCKET type,connect,keywords

Some types:

OPEN, CLOSE, SEND, RECEIVE

Some keywords:

```
   DATA= ...    identifies either a block of data to be sent
                or an area to be used for a receive operation

   LOCAL=YES    tells that a socket call is local to the
                TCP/IP partition and thus no VSE XPCC call
                is required ('internal socket')

   SHORT=YES   reduces the ACKnowledgement meachanism if issued
                in a SEND request. Beneficial for a single query
                over a connection

   CICS=YES     should be used in CICS partitions to use CICS
                GETMAINs (DSA) instead of VSE GETVISes

   WAIT=YES     indicates whether a wait mechanism should be
                incorporated into the SOCKET macro.
                ???
```

For more info and description of the available SOCKET macros in VSE/ESA, refer to

'TCP/IP for VSE/ESA - IBM Program Setup and Supplementary Information', SC33-6601-03 (09/2000)

'TCP/IP for VSE, Programmer's Reference, Rel. 1.4'. 4th edition, 10/99

Socket APIs ...

Ù **Performance Aspects**

„ **Socket APIs**

Note that most of the provided SOCKET implementations require the use of LE with C-runtime.

The most efficient API from a performance point-of-view is the Assembler SOCKET macro interface.

„ **Basic consideration**

When you code your own socket applications, try to communicate as effectively as possible via the sockets: especially, try to SEND and RECEIVE as much data as possible per socket call.

„ **Number of concurrently active sockets**

Be aware that in case of many opened (external) sockets, the amount of CPU-time required for dispatching and searching may increase a lot.
All (external) sockets are currently still chained in a single queue which is searched sequentially.

„ **TCP vs UDP**

Measurements with TCP/IP for MVS sockets have shown that TCP always outperformed UDP:

- less CPU-time overall
- higher throughput
  (loss of UDP packets must be avoided, thus risky when UDP is driven the hard way)

Value of Multiple TCP/IP Partitions

Value of Multiple TCP/IP Partitions

Each TCP/IP copy ('protocol stack') has

- a separate IP address
- a separate host name
- its own set of active, started interfaces (e.g. adapters)
- its own setup of startup parameters

Í **Multiple TCP/IP stacks for functional or performance reasons**

Ù **Functional Reasons**

„ **Separation of workloads**

Include the following aspects

- Availability
- Security
- Buffer pool and priority selections

Separation of Production and Test and/or Education

Separation of production workloads (greater operational flexibility)

„ **Separation of networks (e.g. security)**

- the Internet
- an intranet

Value of Multiple TCP/IP Partitions ...

Multiple TCP/IP Partitions (cont'd)

Ù **Performance Reasons**

„ **Exploit more than 1 engine for TCP/IP: 'Concurrent Dispatch'**

Most of protocol stack related processing is done under a single task (from an operating system view)

Multiple stacks can exploit multiple engines on an n-way (requires VSE/ESA Turbo Dispatcher).

This may be important for TN3270, e.g.

„ **Need of more virtual storage below the line**
(For Telnet alone no more required since Service Pack K)

Before TELNET daemons with POOL=YES were available, and before major areas are moved above the 16M line ...

it was more often required to have >1 TCP partitions (for VS-24 capacity reasons).

Refer to the separate charts on virtual storage capacities for
- Telnet
- FTP
- GPS

„ **Individual Customization**
Usually, it should be possible to find a good compromise e.g. for TN3270 and other concurrent activities.

„ **Separation of TN3270 and FTP/LPR activities**
For higher concurrent FTP or LPR activity, a separate TCP/IP partition may be reasonable, cross linked to the first one.

Refer to 'Mixed TCP/IP Load (TN3270 + FTP)'

TCP/IP for VSE and VM/VSE

TCP/IP for VSE and VM/VSE

Ù **TCP/IP for VM provides similar functions as TCP/IP for VSE**

Ù **Some Aspects**

„ **Both TCP/IPs could communicate via Virtual CTC**

„ **Network ports could be 'shared' between both or partly 'dedicated' to VM or VSE**
Each OSA port consists of several device addresses (CUUs). Sharing is also possible between LPARs.

„ **TCP/IP for VSE 'is closer' to VSE data**

thus
- separate steps or constructs between VM and VSE are not required

„ **Functions you only can do with TCP/IP for VSE:**

- directly get/put data from/to a VSE file (FTP, NFS, ...)
- ...

„ **Reasons to have TCP/IP for VM on top of TCP/IP for VSE:**

- VM/ESA applications with TCP/IP sockets
- Access to VM/ESA files

- VM/ESA as central router for several VSE guests
- Use of TN3270 in VM/ESA for single/multiple VSE/VTAMs (via DIAL)

TCP4VSE Performance View

PART D.

TCP4VSE Performance View

```
For the sake of briefness, sometimes 'TCP/IP for VSE/ESA' is only
referred to here as 'TCP/IP for VSE' or 'TCP/IP'
```

TCP/IP and VSE Partitions

TCP/IP and VSE Partitions

```
      -----------------------------------------------------------
     |TCP/IP partition | VTAM        | CICS            | Batch     |
     |(incl. Daemons,  |             |(a VTAM applic.)|          |
     | Client Mgrs)    |             |                |          |
     |                 |             |                |          |
<===>|==TELNET==DDDDD==|====DDDDD====|<====>          |          |
     |                 |             |                |          |
<====|==GPS=*1==ddddd==|====ddddd====|<=====          |          |
     |                 |             |                | API code |
<===>|==FTP===         |             | API code       |(sockets) |
     |        |        |             |=CICS clients:  |=Batch    |
<===>|==NFS===|        |             |  - FTP         |  clients |
     |        |        |             |  - TELNET *4   | e.g. FTP |
====>|==LPD===|        |             |  - LPR         |          |
     |        |        |             | PING txn       |          |
<====|==LPR===|        |             |                |          |
     |        |    A A |             |            A   |     A    |
     |        |    | | |             |            |   |     |    |
      -----------------------------------------------------------
              A    | |                            |         |
              |    |  ............................          |
           *2|    |             XMOVE                      |
              |     ........................................
              V
             ----
           /      \
          |\      /|
          |  ----  |
          | VSAM   |
          | POWER*3|
          | LIBR   |
          | ICCF   |
           \ SD   /
             ----
```

```
- VTAM is only involved in case of TELNET or GPS

*1 GPS data optionally buffered in VSE library

*2 Access to VSE data is via official interfaces,
   refer to separate foil.

*3 Access to POWER data is via the POWER partition,
   moving data with XPCC in access register mode

*4 For TELNET, any CICS performance monitor shows that the number
   of CICS transactions per message pair (txn) is increased by 2
   (1 CICS-txn in, 1 out).
   Consider that when comparing throughput vs VTAM SNA
```

TCP/IP's Access to VSE Data

TCP/IP's Access to VSE Data

Ù **Summary**

Data	Access via
VSAM	VSAM macros and VSAM code in SVA
POWER	POWER SAS and XPCC (SENDR)
LIBR	LIBRM macro
ICCF	SLI (READ Only) and DTSIPWR in SVA
SD	DTFSD macro (BAM)

Ù **More info (performance related)**

```
- VSAM (ESDS and KSDS):
  GET and PUT, Direct and Sequential
  OPENs are done with 10 index and 10 data buffers

- POWER SAS:
  SAS PUT and SAS GET macros are used.
  CPU-time relevant is the XPCC Send and Reply buffer size,
  which can be up to 64K. TCP/IP uses 32K.

  Per POWER I/O to the POWER Data file, only 1 DBLK block
  is transferred (READ or WRITE).

  -> A bigger DBLK size definitely will help speed up
     transfer of bigger POWER jobs

- LIBRM:
  GET and PUT with BUFSIZE=32000 byte is used

- ICCF:
  SLI and DTSIPWR is used to read members

- DTFSD (Sequential Disk/SAM/BAM) access:
  Per GET and PUT request BLKSIZE bytes are transferred,
  just as the local VSE definition of the file.
```

Ù **These file related macros hold for FTP, NFS, and HTTP**

TCP/IP Virtual Storage Requirements

Virtual Storage Requirements

„ **TCP/IP partition:**

```
31-bit exploitation in the TCP/IP partition started with NFS and
continued with TELNET and other functions.

All TCP/IP GETVIS allocations are tagged with a unique GETVIS
subpool-ID.
```

„ **Other partition(s) with Socket applications:**

```
TCP/IP for VSE/ESA allows socket applications to exploit 31-bit
addressing
             - SOCKET macro
             - BSD/C macros
             - Pre-processor API (EXEC TCP)
```

Shared Storage Aspects

„ **VLA-31:**

C runtime module	CEEEV003 (964K)	Recommended (to avoid FETCHes)

„ **VLA-24:**

TCP connection manager	IPNTCTCP (34K)	Recommended for stability reasons, not performance

```
Do NOT put the TELNET Daemon TELNETD (44K) into the VLA-24.
There is no performance or other benefit. This reentrant phase
is only loaded as a single copy when in partition space.
```

„ **System GETVIS-31:**

```
Refer to chart 'Telnet VS Capacity'
```

„ **System GETVIS-24:**

```
E.g. SOBLOKs (Buffers for external socket requests) <1K
```

TCP/IP for VSE/ESA Startup Job

TCP/IP for VSE/ESA Startup Job

```
* $$ JOB JNM=TCPSTRT,CLASS=7,DISP=L          (F7 is default)
// JOB TCPSTRT
... LIBDEFs etc ...
// SETPFIX LIMIT=900K
// EXEC IPNET,SIZE=IPNET,PARM='ID=0x,INIT=IPINIT0y',DSPACE=3M
/*
/&
* $$ EOJ
```

„ **VSE partition size**

a) Before Service Pack J (02/99):

í **Let it end 1M above the 16M line**

just to be able to fully exploit 24-bit private space: Avoid 31-bit eligible pgms/areas below 16M.

to provide some GETVIS-31 for system functions, including space for VSAM buffers.

b) Sevice Pack J and later:

í **Specify sufficient space above the line**

Be generous and provide enough space for all areas moved above the line (refer to separate chart).

Start e.g. with 20M partition size

Non-used virtual storage does only occupy VSIZE, so you really can afford to be generous

You may monitor GETVIS via 'GETVIS part-ID'.

í **Add 3M on top for NFS (any Service Pack)**

to provide enough GETVIS-31 for NFS functions
Just for starting, maybe reduce later if you want

TCP/IP for VSE/ESA Startup Job ...

TCP/IP for VSE/ESA Startup Job (cont'd)

„ **SETPFIX LIMIT=**

All I/O interface drivers are PFIXed, including I/O buffers. This is required, since due to also supporting unknown devices, the CCW translation is done by TCP/IP.

All task control blocks are PFIXed, in order to avoid page faults in this performance relevant code.

í **Start e.g. with SETPFIX LIMIT=900K**

- to cope for all adapter types/configs/high loads
- just to be on the safe side.

í **Monitor actual requirements via MAP REAL**

But specifying a higher value than required does not harm.

„ **Type of VSE partition**

Since TCP/IP for VSE/ESA is up for long times ...

- it is a long lasting VSE job step

í **A VSE dynamic partition is very well suited**

TCP/IP for VSE/ESA Startup Job ...

TCP/IP for VSE/ESA Startup Job (cont'd)

„ **SIZE=IPNET**

Should be as indicated, to give as much storage as possible to partition GETVIS-24.

Note

- There is only code contained in the partition program area
- NFS code is being loaded into GETVIS (currently GETVIS-24)
- Leave about 1M GETVIS-24 for the (old) command processor (no more required/applicable for new command processor)
- Do not increase dynamic space GETVIS beyond shipped values (reduces GETVIS-24)

„ **IPINIT0x**

Contains all relevant TCP/IP parameters discussed later on

Can be setup with the TCP/IP for VSE/ESA Configuration Support Tool (on Windows 3.1 or higher, or OS/2).

„ **DSPACE=3M**

This is the maximum size of the dataspace used by VTAM for this VTAM application.

It is better to specify DSPACE, otherwise its default SYSDEF DSPACE,DFSIZE=mM has to be found out, or even may be too low.

„ **DSPACE parameter for VTAM startup**

The DSPACE parameter in the VTAM startup job specifies the maximum size of VTAM's own dataspace.
With heavy TCP/IP traffic, up to 6M and more may be required

TCP/IP for VSE/ESA Dispatch Priority

TCP/IP for VSE/ESA Dispatch Priority

General

A high TCP/IP for VSE partition priority improves not only TCP/IP performance/throughput, but also may be required to avoid time critical situations in TCP/IP.
Reasonable settings may also depend on

- Type of TCP/IP application (TN3270, FTP, FTPBATCH)
- Mix of TCP/IP applications (in same TCP/IP partition)
- Potential impact on other loads (TCP/IP and others)
- Dispatcher type (TD allows PRTY SHARE settings and n-ways)

Rough Recommendations:

1. **Select PRTY sequence (low to high):**

   Batch, CICS, TCP/IP, VTAM, POWER

2. **A 2nd TCP/IP partition is highly recommended, if, besides Telnet,
   concurrent FTP activity (other than FTPBATCH)
   or if LPR/LPD applies**

TCP/IP for VSE/ESA Dispatch Priority ...

TCP/IP for VSE/ESA Dispatch Priority (cont'd)

```
  'Low priority' (or SHARE)                'High priority'
 ==========================================================>
 a       b         c          d           e        f      g

... ------------ ....... -------------- ........ -------- ...
   |            |       | CICS          |        | VTAM   |
   | BATCH load |       | Online load   |        |        |
   |            |       |               |        |        |
... ------------ ....... -------------- ........ -------- ...


Reasonable PRTY selections:

 TCP/IP partition with - TN3270 only:          d e f g
                       - FTP only:           c d e
                       - both:                 d e f


- 'a' to 'g' are priority 'positions'
  (b, d, f stand for 'in same Partition Balancing group as the
   pertinent load'; note that only 1 PB group is allowed)

- VSE/POWER not shown here.
  Separate priority considerations may apply, already w/o TCP/IP

- Guaranteed share of CPU resource is only provided by the
  Relative VSE Shares of the TD: PRTY SHARE,Fx=nnnn

- Selection of individual TCP/IP partition priorities is also
  influenced by need for concurrent batch throughput

- TCP/IP partition priority should be as high as required,
  in order to avoid e.g. unnecessary retransmissions
```

TCP/IP for VSE/ESA Dispatch Priority ...

TCP/IP for VSE/ESA Dispatch Priority (cont'd)

TN3270 (only) general rule

If for an online transaction load different partitions are required for processing, usually it is best to give HIGHEST priority to that partition of this set, which has LOWEST CPU consumption.

First experiences with TN3270-only workloads have shown that response times only hardly suffered when the TCP/IP partition even had lower priority than the related CICS partition.

Putting the TCP/IP partition (F7) into the same partition balancing group as CICS in Fx, was a reasonable compromise:

```
        e.g.    PRTY ..., 'Fx'=F7,F3,F1
```

Note that with slower or unreliable networks, TCP/IP should get a priority as high as possible.

FTP (only) general rule

For FTP, a tradeoff between potentially higher transfer rates and lower impact on other loads must be chosen.

TN3270 and concurrent FTP in 1 TCP/IP partition

Parameter selection itself may be a compromise, partition priority also.

Separate TCP/IP partitions for TN3270 and FTP

Refer to the foil 'Mixed TCP/IP Load'

Mixed TCP/IP Load (TN3270 + FTP)

Ù **High concurrent FTP (or LPR/LPD) activity may/will impact e.g. TN3270 response times**

```
        via  - processor (CPU-time)
             - DASD access
             - high network/link/adapter utilization
Both type of loads are using the same resources
```

Ù **Conflicting Targets**

```
 - Make FTP as fast as possible (especially single stream)
 - Have a small impact on other concurrent loads
```

Ù **Potential Solutions**

„ **Separate the files and DASDs**

Normally not feasible, since often the same data are transferred as used by other Online loads (e.g. TN3270)

„ **Do NOT allow huge FTPs during prime shift**

E.g. limit the number of FTP daemons via COUNT=0x.
Not THE solution for all cases

„ **Vary MAX_BUFFERS (Service Pack 'G')**

Use MAX_BUFFERS=1 to limit single FTP session buffer usage.

„ **Separate adapters may help**

if FTP bandwidth to be limited on a higher level

„ **Make it controllable by the system programmer**

VSE cannot e.g. THROTTLE a device like VM can.

Best would be within TCP/IP, but FTP uses the SAME TCP/IP stack as TN3270

„ **A separate TCP/IP partition for FTP**

Refer to next foil

„ **Use FTPBATCH in a separate partition**

(Service Pack J and up). Refer to separate chart.
Reduces the need for a separate 'FTP TCP/IP partition'.

Separate TCP/IP partition for FTP

Separate TCP/IP partition for FTP

```
             ID=00                ID=01              -----
           -----------         ------------        /     \
          |           |       |            |      |\     /|
          |   TN3270  |       |   FTP  A   |<---->| ----- |
           -----------         --------.---       |  DASD |
          |   TCP     |       |   TCP  .   |       \     /
           -----------         --------.---        -----
  to/from|   IP       |       |        .   | to/from
  TCP/IP |   ........|<------>|<..IP....   | TCP/IP
  network ---.-------  LINK01  ------------  network  (best!)
 <------>|....        |       |            | <------->
           -----------         ------------
            primary             secondary
```

Ù **Setup**

„ **Set up a 2nd TCP/IP partition (for FTP)**

with separate IPINIT01 initialization member
and 1 additional IP-address

„ **You may leave the network connection to outside VSE in 1st TCP/IP partition and cross-partition-connect both**

```
via    DEFINE LINK,ID=LINK01,TYPE=IPNET,SYSID=01
       DEFINE LINK,ID=LINK01,TYPE=IPNET,SYSID=00
```

or (recommended):
let each partition have its own network access

Ù **Benefits**

Refer to next chart

Separate TCP/IP partition for FTP ...

Separate TCP/IP partition for FTP (cont'd)

```
            ID=00                ID=01             -----
         -----------          ------------        /     \
        |           |        |            |      |\     /|
        |   TN3270  |        |   FTP  A   |<---->| ----- |
         -----------          --------.---      |  DASD |
        |   TCP     |        |   TCP  .   |       \     /
         -----------          --------.---         -----
 to/from|   IP      |        |        .   | to/from
 TCP/IP |   ........|<------>|<..IP....   | TCP/IP
 network ---.-------  LINK01  ------------ network  (best!)
<------>|....       |        |            | <------->
         -----------          ------------
          primary              secondary
```

Ù **Setup**

Refer to last chart

Ù **Benefits**

„ **Different VSE priorities possible for TN3270 and FTP**

Priorities are even changeable on the fly

„ **For TD environments**
- VSE Relative Shares may help on top
- FTP even may run on another engine

> FTP only consumes remaining CPU power
  - only when no higher priority load is dispatchable,
    i.e. does not overduly hurt TN3270 (or even CICS SNA) load

In case the secondary TCP/IP has no own network access, some direct impact of FTP on TCP/IP remains, since the IP-layer in the first TCP/IP partition is common.

Also higher CPU-time for FTP required (see FTP results with Gateway TCP/IP)

Batch FTP from a Separate VSE Partition

Batch FTP from a Separate VSE Partition

Ù **// EXEC FTP**

FTP initialization is done from a VSE batch partition

No performance related benefits

Only FTP initialization and termination runs in a separate batch partition.

Ù **// EXEC FTPBATCH (Service Pack J, 02/99)**

A special FTP program for autonomous files.

Performance related benefits

„ **Potential exploitation of >1 engine of an n-way**

„ **Separate File-I/O routine used per FTP**

Single thread TCP/IP File routine in TCP/IP partition not used/blocked

„ **Control of FTP batch CPU dispatch priority**

- via PRTY setting with Std Dispatcher
- via PRTY (and TD Relative Shares) on 1- or n-way

„ **Move of data between batch and TCP/IP partition using access registers**

„ **Separate counting of CPU-time and file I/Os via VSE JA**

File access of other FTPs is done in TCP/IP partition

„ **Info on '#bytes sent/received so far'**

- via MSG to partition
- via setup ('timed')

31-bit Exploitation in Serv.Pack J

31-bit Exploitation in Service Pack J (and up)

Ù **Many control blocks and buffers moved above the line**

„ **TN (Telnet) blocks (partly)**

Telnet definitions. Subpool TNBLOK

„ **IBBLOCKs**

Buffers for transfer of data (TCP, UDP, ARPs...). Subpools IBBKxxx.

„ **Telnet buffers**

POOL=YES and POOL=NO

Refer to SET TELNETD_BUFFERS description. Subpool TBBLOK.

„ **FTP transfer buffers**

Refer to SET TRANSFER_BUFFERS description. Subpool TBBLOK.

Í **Major Virtual Storage Constraint Relief (VSCR), especially beneficial/required for Telnet**

Í **Start with e.g. 20M partition size**

If real storage is no problem ...

select POOL=NO for TELNET
(2x8K TELNET buffers per daemon, in GETVIS-31)

TCP/IP for VSE Defaults

Product/Shipped Defaults for TCP/IP for VSE/ESA

Please distinguish between the following type of defaults:

„ **Product defaults**

Used whenever

- NO assignment of a value is explicitly specified in the IPINIT0x source book ('assembled defaults')

and

- also was not explicitly set via a SET command

„ **Shipped defaults**

Values specified in the shipped IPINIT00.L source book in PRD1.BASE.

Changes can be done by the user, be it
- via any editor or
- via the Configuration Dialog.

The shipped startup values usually represent a good starting point for being used, except there are good reasons for a change, based on specific loads or configurations.

In some instances, the IBM shipped values may differ from the values shipped by CSI.

For NFS, shipped defaults are contained in the NFSCFG.L member, and in general are identical with the product defaults.

In the following command descriptions, both the product default, and the IBM shipped default are cited.

TCP/IP for VSE Commands

TCP/IP for VSE Commands

MTU values are defined in the following DEFINEs

They here only apply to outbound traffic

Ù DEFINE ADAPTER

```
DEFINE ADAPTER,LINKID=...,NUMBER=...,TYPE=...,MTU=maxunit,
FRAGMENT=...,MODIFY ...
```

```
FRAGMENT=YES|NO  YES allows that fragmented IP datagrams are being
                 sent out by TCP/IP for VSE/ESA
                 (which always can receive fragmented datagrams).

                 NO is default and avoids that the receiving side
                 can get fragmented IP datagrams
                 (which it may not be able to handle).
                 NO should be used whenever possible.

  MTU=xxxx       (default is 1500). 576 is min, optimal values may
                 be higher, but any selected value must be suppor-
                 ted by every device attached to the adapter
```

Ù DEFINE LINK

```
DEFINE LINK,ID=...,TYPE=...,DEV=...,MTU=xxxx,FRAGMENT=...,
...
```

```
TOKEN-RING:

  FRAME=xxx     max frame size of T-Ring adapter, default=2052
                (512, 1500, 2052, 4472, 8144, 11407, 17800)

CLAW (Common Link Access for Workstations) only:

  INFACTOR=infact,OUTFACTOR=outfact,
  INBUFFERS=inbuff,OUTBUFFERS=outbuff

 with
  infact  = size of the  input buffers
  outfact = size of the output buffers 1 .. 8 (K), default=4,

  inbuff  = #input  buffers chained, 1..16, default =4, VM needs 1
  outbuff = #output buffers chained, 1..16, def=recomm=4

LINK to other TCP/IP partition in same VSE: TYPE=IPNET
```

TCP/IP for VSE Commands ...

Ù DEFINE TELNETD

```
DEFINE TELNETD,ID=..., ...,POOL=YES|NO
```

```
POOL=NO (default)  Each TELNET daemon gets 2x8=16K buffers assigned
                   when activated (8K for in-, 8K for outbound)
                   for exclusive use.
                   In GETVIS-24 or GETVIS-31 (ServicePack J)


POOL=YES           The TELNET daemon uses a 2x8=16K buffer from the
                   TELNETD buffer pool when required
                   (8K for in-, 8K for outbound).

                   Shared buffers are allocated immediately when
                   their number is defined via SET TELNETD_BUFFERS.
                   In GETVIS-24 or GETVIS-31 (ServicePack J)
```

Ù DEFINE FILE

```
DEFINE FILE,TYPE=..,DLBL=..,LRECL=..,BLKSIZE=..,NFSTIMER=..
```

This command adds a VSE file to the TCP/IP file system, for use by FTP and NFS.

```
 LRECL           Logical record length, should be consistent with
                 the file definition.
                 Default value is 80 byte.

 BLKSIZE         Physical blocksize, should be consistent with the
                 file definition (no default value exists).

 The order of precedence for LRECL and BLKSIZE is:
          DLBL, DEFINE FILE, and then the FTP SITE command.

 What values in DLBL??
 What is the performance impact if BLKSIZE bigger??

 NFSTIMER        Time interval (in sec) for NFS to keep directory
                 info of that file in its NFS directory cache
                 (Refer to NFS part).
```

TCP/IP for VSE Commands ...

Ù PING

```
PING ipaddr
```

ECHOes to a specified IP address and gives individual round-trip times for 5 successive PINGs in msec.

Ping uses specific ICMP messages (Echo and Echo Reply), which are directly put into IP datagrams (ICMP must conceptually be implemented on the IP level).

TCP/IP for VSE has PING implemented

```
      - as CICS transaction   PING ipaddr
      - as TCP/IP command     PING ipaddr
      - as Batch client       // EXEC CLIENT,PARM='AAPPL=PING'
                              SET HOST=ipaddr
                              PING
```

In TCP4VSE, it however does not allow to also specify the size of the packet(s) to be sent and measured.

PING can be used ...
- for functional purposes (to test connection)
- (very very roughly) for performance purposes

Usually, these round-trip times for IP datagrams very roughly correspond to the average time until an IP datagram has caused an ACK to be sent back from the remote TCP. In any case, it is a Snapshot and also done with another protocol!

Usage hints:

For best-can-do determination of PING times ...

- Do multiple PINGs.
  First PING may be much longer (if an ARP request was needed since no MAC address was available). Don't use it, if so.

  Use an average value, since PING times vary with

  - the actual traffic on the network
  - the priority of TCP/IP in VSE
    (and the current processor situation)
  - the priority of TCP/IP on the other side
  - the route(s) taken

TCP/IP Performance Related Parameters

TCP/IP Performance Related Parameters

The following tables show those settings in TCP/IP for VSE which are performance relevant, together with the type of TCP/IP activities a parameter has influence on.

Settings for traces or debugging are not inluded.

	Scope of TCP/IP Activity				
TCP/IP Parameter/setting	Any	Outbnd. only	Inbound only	TN3270 Out+In	FTP Out+In
DEFINE ADAPTER|LINK MTU		X			
TELNETD POOL				X	
SET ALL_BOUND	X				
DISPATCH_TIME	X3				
REDISPATCH	X3				
ARP_TIME	X				
REUSE_SIZE	x				
FULL_SCAN	X				
GATEWAY	x				
CHECKSUM	x4				
SET MAX_SEGMENT			X1		
WINDOW_DEPTH			X1		
CLOSE_DEPTH			X4		
WINDOW_RESTART			X1		
SET RETRANSMIT		X1			
FIXED_RETRANS		x1			
WINDOW		X1			
ADDITIONAL_WINDOW		x1			
SET SLOW_START		x4			
SLOW_RESTART		x4			
SLOW_INCREMENT		x4			
SET TELNETD_BUFFERS				X2	
TRANSFER_BUFFERS					X
MAX_BUFFERS					X

X1 Only for TCP loads (includes FTP, but not NFS)
X2 Only for POOL=YES TELNET daemons/sessions
X3 Parameter influence reduced since SPack K
X4 New in TCP/IP 1.4

TCP/IP Performance Related Parameters ...

TCP/IP Performance Relevant Parameters (cont'd)

Scope of NFS Activity

NFS Parameter/setting	Any NFS	NFS MOUNTs	NFS Dir READs	NFS READs	NFS WRITEs
DEFINE FILE NFSTIMER			X		
DIRCACHESIZE DIRGROUPSIZE		X	 X		
READCACHESIZE READCACHETIME				X X	
VSAMTABLESIZE WAKEUPTIME WRITECACHETIME MAXREQUESTS	 X X				X5 X

X5 NFS VSAM WRITEs only
- No individual tuning parameters for LPR/LPD and HTTP

Some Perf. Related SET Commands

Some Performance Related SET Commands

All timer values are in multiples of 1/300 sec.

Ù SET ALL_BOUND (all loads)

```
SET ALL_BOUND=atime
```

Maximum idle time, similar to CICS ICV. Default is 9000 (30 sec). Shipped 'default' is 30000 (100 sec)

This value is 'only' to ensure that no TCP/IP work is available. There is no risk to set it higher for VSE/ESA, since this should not occur due to a POST mechanism used.

If the value is too low, unnecessary CPU overhead is caused

Ù SET DISPATCH_TIME (all loads)

```
SET DISPATCH_TIME=dtime
```

Maximum time-slice a single TCP/IP pseudo task can get, before being interrupted in favor of another TCP/IP pseudo task. Default is 6 (0.02 sec). Shipped 'default' is 30 (0.1 sec)

CPU-time impact is similar as for ALL_BOUND.

For FTP this time should be high (avoid CPU-time overhead). For TELNET and any interactive TCP/IP use, a lower value may give better and more consistent response times.

Settings have a much reduced impact since new dispatching in SPack K and above.

Ù SET ARP_TIME (all loads)

```
SET ARP_TIME=arptime
```

Amount of time, before the ARP table is being rebuilt. Default is 90000 (5 min), should not be smaller.

Some Perf. Related SET Commands ...

Some Performance Related SET Commands (cont'd)

Ù SET PULSE_TIME (all loads)

```
SET PULSE_TIME=arptime
```

Amount of time a connection is allowed to be idle, before checked ('dead' connection). Default is 18000 (1 min), should not be smaller.

Ù SET REDISPATCH (all loads)

```
SET REDISPATCH=rdtime
```

Stall(=Wait) interval to re-dispatch pseudo-tasks. This value determines the time interval after a non-interruptible TCP/IP pseudo task is again being tried to be interrupted. Default is 1 (1/300 sec). Shipped 'default' is 10 (1/30 sec)

Too high values may cause erratic response times, too low values will increase CPU-time by too frequent unsuccessful trials.

Redispatch counter:

The redispatch counter in the SET RECORD=ON display shows how often a certain task was redispatched, since it could not be interrupted at the end of its time slice (in 'fragile' state).

Due to VTAM services, TELNET daemons tend to show higher redispatch counters.

Settings have a much reduced impact since new dispatching in SPack K and above.

Some Perf. Related SET Commands ...

Some Performance Related SET Commands (cont'd)

Ù SET REUSE_SIZE=nn (all loads)

```
SET REUSE_SIZE=nn
```

REUSE_SIZE controls the depth of the reusable control block queues for IBBLOCKs (new in Sevice Pack J).

nn is the number of free control blocks of each fixed size that are retained for reuse (i.e. not FREEVISed), default is 10.

Before J, a high value was used (implicitly), thus potentially saving GETVIS/FREEVIS requests, at cost of virtual storage below (now above) the line.

This effect depends on the amount of data transferred, and is lower for Telnet than for FTP mass transfer of data. For Telnet runs shown, more than 20 did not show measurable CPU-time benefit.

Using the default of 10 for Telnet looks OK, more does not harm.

Ù SET FULL_SCAN (all loads)

```
SET FULL_SCAN=num
```

Determines how often TCP/IP is forced to do a dispatch of the full queue of all ECBs (tasks) rather than doing 'fast dispatches' on the same/similar dispatch level.

'num' is the max number of dispatches allowed before such a full dispatch cycle must be done.

Default is 10.

A high FULL_SCAN value reduces CPU time while creating small delays for new work.

A low FULL_SCAN value increases CPU-time while being very responsive for new work.

Some Perf. Related SET Commands ...

Some Performance Related SET Commands (cont'd)

Ù SET MAX_SEGMENT (any TCP inbound load)

```
SET MAX_SEGMENT=num
```

Is the maximum size of TCP data and thus limits the max. accepted TCP segment size (to tell to remote hosts only). Range is 576 .. 32684, default is 32684. 'RECEIVE MSS' in QUERY ALL display

It is recommended to use the max. MTU-size for the adapter/link minus 40 bytes, except for functional problems with Token Rings.

A MAX_SEGMENT size of 576 would cause a maximum frame size of 576+40=616 (if FRAGMENT=NO and IP header is 20 byte)

TCP/IP for VSE/ESA

- uses MTU size to limit outbound traffic only (max. frame size)
- always could provide sufficient buffering to receive the largest datagram valid to the protocol.

Ù SET WINDOW (any TCP inbound load)

```
SET WINDOW = wsize
```

RECEIVE window size (#bytes a sender may send to VSE TCP/IP before he needs an ACK). Default is 8192 (bytes). Shipped 'default' is 4096, max. value is 64K.

A high value may slow down detection of a lost connection, a low value may cause delays due to waiting for ACKs.

This value is used in order to set the SEND window size at the remote TCP/IP, when a session is established.

It holds for inbound transfers.

Some Perf. Related SET Commands ...

Some Performance Related SET Commands (cont'd)

Ù SET WINDOW_DEPTH (any TCP inbound load)

```
SET WINDOW_DEPTH=wd
```

Number of data segments or IBBLOKs (Inbound Buffer Blocks, 256 + IP_datagram_size) which can be concurrently queued inbound in TCP, before a sender is notified by indicating a current window size of 0 ('window shutdown'). Default is 30. 'WINDOW DEPTH' in QUERY SET display. TCP/IP 1.4 only gradually reduces the window size, as compared to 1.3.

'DEPTH MODE' is shown in the QUERY IPSTATS display. It is entered whenever the number of data segments exceeds WINDOW_DEPTH, and is exited when smaller than WINDOW_RESTART.

Ù SET WINDOW_RESTART (any TCP inbound load)

```
SET WINDOW_RESTART=wr
```

Number of data segments queued at which the TCP window is re-opened for inbound transmissions. Default is 10.

'WINDOW RESTART' in QUERY SET display

Ù SET CLOSE_DEPTH (any TCP inbound load)

```
SET CLOSE_DEPTH=num
```

This value determines how many TCP segments are still accepted, in spite of a fully closed window.

Default is CLOSE_DEPTH=10.

Some Perf. Related SET Commands ...

Some Performance Related SET Commands (cont'd)

Ù SET RETRANSMIT (any TCP outbound load)

```
SET RETRANSMIT=rttime
```

Time interval before retransmission of unacknowledged 'packets'. Default is 50 (0.166 sec). Shipped 'default' is 100 (0.33 sec)

An optimal value is

- not too low to avoid unnecessary retransmits (network link(s) are slow, but reliable)
- not too high to cause unnecessary delays in case of required retransmissions (network link(s) are not reliable)

TCP/IP applies a dynamic/adaptive retransmission concept for each individual TCP connection, using this value as a starting point.

Starting with Service Pack I, this concept can be overruled by a new parameter setting:

Ù SET FIXED_RETRANS (any TCP outbound load)

```
SET FIXED_RETRANS = ON|OFF
```

ON forces that the RETRANSMIT value is not dynamically adjusted.

In case of too many 'retransmissions' done by TCP/IP, you may switch from the default (OFF) to ON (but consult CSI Technical Support before).

Function was added in Service Pack I.

Some Perf. Related SET Commands ...

Some Performance Related SET Commands (cont'd)

Ù SET SLOW_START (any TCP outbound load)

```
SET SLOW_START=num
```

Determines the 'decision interval' for the slow start mechanism, of highest importance for FTP.

Every num sucessfully transferred outbound TCP segments, TCP/IP determines whether the outbound transfer policy/value for a specific TCP connection is to be adjusted, except it would be in a pausing phase defined by SLOW_RESTART.

Default is SLOW_START=10.

Detail info can be obtained in debug situations via 'SET DIAGNOSE=PERFORM': e.g. 'Max. window achieved'.

Function was implemented in TCP/IP 1.4. Refer to SLOW_INCREMENT and SLOW_RESTART.

Ù SET SLOW_INCREMENT (any TCP outbnd load)

```
SET SLOW_INCREMENT=num
```

Determines how aggressively TCP/IP adjusts the value for the slow start mechanism at a decision interval:

A new TCP connection is always started with 2 segments, before TCP/IP waits for an acknowledgement. At a decision interval, this value is being INcreased by SLOW_INCREMENT, if no retransmission occurred and if the foreign window allows.

If a retransmission occurred, this value is DEcreased by SLOW_INCREMENT.

Default is SLOW_INCREMENT=1.

So, there will be an INcrease until retransmission, a DEcrease until 'clean', and then a pause for SLOW_RESTART decision intervals, see below.

Some Perf. Related SET Commands ...

Some Performance Related SET Commands (cont'd)

Ù **SET SLOW_RESTART (any TCP outbound load)**

```
SET SLOW_RESTART=num
```

```
This value determines how many decision intervals a
'no-change period' lasts, i.e. the adjustment algorithm
pauses, after 'no retransmission' was achieved.

Default is SLOW_RESTART=10.
```

Ù **SET ADDITIONAL_WINDOW (any TCP outbound)**

```
SET ADDITIONAL_WINDOW= bytecnt
```

```
This value allows to avoid the 'Silly Window Syndrome' (SWS).
SWS may occur if a TCP/IP partner host signals too small
number of bytes which are freed in his formerly closed window.

Restart after a windowsize=0 from the partner is only done,
when the advertised window of the partner is
     > 80% of max_window + ADDTL_WINDOW

Default is 100 (byte).

Service Pack L allows SET DIAGNOSE=SWS, which gives hints
what to use in case of such a problem.
```

Some Perf. Related SET Commands ...

Some Performance Related SET Commands (cont'd)

Ù **SET TELNETD_BUFFERS (TELNET only, in/out)**

```
SET TELNETD_BUFFERS=numtd
```

```
Number of 16K buffer (8K per direction) in the TELNETD buffer
pool. Used only for TELNET daemons defined with POOL=YES.

Default is 20, appropriate for at least 100 TELNET daemons.

Since buffers are only used when actual data transfer occurs,
     - this number can and should be much lower than the
       number of POOL=YES defined TELNET daemons
     - any number greater than that is waste of virtual storage
       below the line (before Service Pack J)

Rules of Thumb:  Use 15 TELNETD buffers for 10 txn/sec
                 or 2.5 to 7.5 buffers for 100 terminals
```

Ù **SET TRANSFER_BUFFERS (FTP only, in/out)**

```
SET TRANSFER_BUFFERS = numt
```

```
Total number of 32K transfer buffers allocated to the FTP buffer
pool (above the line) shared by all FTP daemons.
Default is 10, shipped 'default' is 20 if FTP used.

See MAX_BUFFERS for trade-offs
```

Ù **SET MAX_BUFFERS (FTP only, in/out)**

```
SET MAX_BUFFERS=numx
```

```
Limits the number of 32K transfer buffers available to an
individual FTP daemon. The range is 1... 65535, 4 is default.
 - Do NOT specify 0, will fail, though currently accepted.
 - The FTPBATCH command is : SET BUFFMAX=numx

More buffers can temporarily compensate a high FTP transfer
rate (e.g. via CTCA) vs a lower DASD speed.

Too many buffers may
 - limit concurrent data move in and out of the transfer buffers
 - even reduce overall data rate
 - cause VSE paging in very extreme cases
```

CPU-time Overhead of other SETs

CPU-time Overhead of other SETs

Ù **SET SECURITY**

```
All SECURITY defaults are OFF

 SET SECURITY=OFF|ON     Verify for user ID and password
                         -> negligible performance impact

 SET SECURITY_IP=OFF|ON  Check IP addressing, every time a
                         connection is established for TCP
                         -> small performance impact

 SET SECURITY_ARP=OFF|ON Check H/W address for inbound requests
                         -> higher performance impact
```

Ù **SET DEBUG**

```
 SET DEBUG=OFF|ON|FULL   Controls how much internal debug info
              |PRINTER   is displayed on console (or SYSLST).
                         Of special value during initialization.
                         -> CPU-time overhead, highest for FULL
                            (default is OFF)
```

Ù **SET DIAGNOSE**

```
 SET DIAGNOSE=OFF|       Controls production of diagnostic info
                         for specific functions.
             STORAGE|    Allocations of all IBBLOKS
             SWS|        Info to diagnose Silly Window Syndrome
                         (window not fully re-opened by partner)
                         -> CPU-time overhead (Serv. Pack L)
             PERFORM     See separate charts.
```

Ù **SET MESSAGE**

```
 SET MESSAGE xxx=ON|..   Controls production (type and target)
                         of messages.
                         TCP/IP for VSE does issue only seldomly
                         console messages when up
```

Ù **DEFINE TRACE**

```
 DEFINE TRACE,ID=...    Starts tracing into memory for a speci-
                        fied IP address or all incoming traffic
                        -> bigger CPU-time overhead
```

IPINIT Excerpts for Performance

IPINIT Excerpts for Performance

The following are some lines of an IPINIT member for TCP/IP 1.3. Here, just performance relevant lines or parameters are shown.

It may be good practice, to specify values even with their product or shipped defaults, to be aware of their existence/relevance.

For details, refer to foils explaining the individual commands.

```
*////////////////////////////////////////////////*
*           Define the constants                 *
 ...
* Next 2 lines are to assure no waste of virtual
* storage in case these buffers are not needed
*
SET TELNETD_BUFFERS  = 0
SET TRANSFER_BUFFERS = 0
* ====== For all TCP/IP Activities ============= *
*
SET ALL_BOUND        = 30000
SET DISPATCH_TIME    = 30
SET REDISPATCH       = 10
*
* ====== For all TCP Inbound Activities ======== *
*
SET MAX_SEGMENT      = 32684
SET WINDOW_DEPTH     = 30
SET WINDOW_RESTART   = 10
*
* ====== For all TCP Outbound Activities ======= *
*
SET RETRANSMIT       = 100
SET WINDOW           = 4096
*
* ====== For TELNET_3270 Only ================== *
* Comment out next line if no TELNET used!
SET TELNETD_BUFFERS  = 20
*
* ====== For FTP Only ========================== *
* Comment out next line if no FTP used!
SET TRANSFER_BUFFERS = 20
SET MAX_BUFFERS      = 6
 ...
SET DEBUG  = OFF
SET RECORD = OFF
SET DIAGNOSE = OFF
*------------------------------------------------*
*           Wait for VTAM Startup                *
WAIT     VTAM
...
```

IPINIT Excerpts for Performance ...

```
  ...
 *------------------------------------------------*
 *           Define the Communication Links       *
 * DEFINE LINK, ... ,MTU=1500,FRAGMENT=NO, ...
   ...
 * DEFINE ADAPTER, ... ,MTU=1500,FRAGMENT=NO, ...
   ...
 *------------------------------------------------*
 *           Define Routine Information           *
 * DEFINE ROUTE,...
 *------------------------------------------------*
 *           Define TELNET Daemons                *
 DEFINE TELNETD, ... ,POOL=YES, ...
 *------------------------------------------------*
 *           Define FTP Daemons                   *
 DEFINE FTPD,..., COUNT=0x
 *------------------------------------------------*
 *           Line Printer Daemons                 *
 DEFINE LPD,...,LIB=library
 *------------------------------------------------*
 *           Automated Line Printer Client        *
 DEFINE EVENT,...
 *------------------------------------------------*
 *           Setup the File System                *
 DEFINE FILESYS,LOCATION=SYSTEM,TYPE=PERM
 *
 DEFINE FILE,PUBLIC='IJSYSRS',DLBL=IJSYSRS,TYPE=LIBRARY
  ...
 *------------------------------------------------*
 *           Define Gopher Daemons                *
 DEFINE GOPHERD, ...
 *------------------------------------------------*
 *           Define HTTP Daemons                  *
 DEFINE HTTPD, ...
 *------------------------------------------------*
 *       Define NFS Daemon (after DEFINE FILEs)   *
 DEFINE NFSD,CONFIG=NFSCFG, ...
 *------------------------------------------------*
 *           Setup member NETWORK.L               *
 INCLUDE NETWORK,DELAY
 *////////////////////////////////////////////////*
```

NOTE:
TELNETD shared buffers and TRANSFER buffers for FTP are allocated directly when SET in the startup, even before any POOL=YES TN daemon or FTP is defined.

===> These statements should be made inactive if not required

Some Informational Display Commands

Some Informational Display Commands

Ù **QUERY VERSIONS**

Displays the TCP/IP version and current maintenance level

Ù **QUERY ALL**

Display all available info (optionally on SYSLST).
Very voluminous, use more specific QUERY commands instead.

Ù **QUERY SET**

Display current setting of all values that can be set via SET

Ù **QUERY STATS**

Display overall operational statistics:

```
- FTP/Telnet daemons: Current/max active, max. active buffers
- LP/HTTP/Gopher daemons: #daemons, #sessions, #requests
- TCP inbound rejections
- FTP Files/bytes sent/received
- Telnet bytes sent/received
- TCP/UDP/IP bytes sent/received
- Received Blocks     Total
                      Inbound Datagrams
                      Non-IP     (should be 0)
                      Mis-Routed (should be very small)
- Transmitted Blocks  Total
                      Outbound Datagrams
```

These statistics are also displayed at TCP/IP shutdown.
They cannot be reset during TCP/IP operation.

Ù **QUERY TASKS**

Display all currently active TCP/IP for VSE (pseudo) tasks with dispatch counts

Ù **QUERY CONNS | CONNECTION,IPADDR=addr**

Displays all active connections and/or connection data, plus
- Maximum Segment Sizes (MSS) for SEND, and for RECEIVE
- Maximum Window that has occurred thus far
- Number of segments/datagrams in/out

Use QUERY CONNECTION, IPADDR=... to check the actually used MSS sizes!

Some Informational Display Commands ...

Some Informational Display Commands (cont'd)

Ù **QUERY LINKS**

Displays the status of all links

Ù **QUERY ACTIVE,TYPE=...**

```
QUERY ACTIVE,TYPE=...  Displays all active items

        TELNETD     all active Telnet daemons
        ...
```

Ù **QUERY TRACES**

Lists all traces in progress. No trace should be active for optimal storage use and lowest CPU-time.

Ù **QUERY ISTATS**

Display internal info (TCP/IP dispatching statistics).
Available since Service Pack J, to be interpreted by CSI.

```
Number of Dispatches: Total
                      Active
                      Fixed
                      Quick
                      Persistent
                      Passed
                      Complete
```

Refer to description in SET RECORD=ON, which displays this info on a task basis

Some Informational Display Commands ...

Ù **QUERY IPSTATS**

Display statistics on individual connections, upon request on the console or at TCP/IP shutdown on SYSLST

```
IP ADDRESS xx.xx.xx.xx
PERFORMANCE INFORMATION
- OVERALL
  #connections, max. turnaround/depth/window
- SWS MODE
  time, count
- DEPTH MODE
  time, count
- RETRANSMIT MODE
  times in mode, count
APPLICATION INFORMATION
- FTP/HTTP/TELNET
  connections, inbound/outbound files/file bytes
TRANSFER INFORMATION
- IP/TCP/UDP
  info on datagrams and bytes (IN/OUTBOUND)
```

Data are cumulative and more selective than e.g. those from SET DIAGNOSE=PERFORM.
No direct information is included regarding IP fragmentation or reassembly.

Seldom Used Displays and Traces

Ù **SET TRAFFIC=ON   (any inbound load)**

```
SET TRAFFIC=FULL|OFF|ON  Allows to control how non-IP traffic
                         is handled

     FULL     No traffic is discarded at IP level.
              Allows to trace non-IP data
              via DEFINE TRACE for 1 or all IP addresses
              (big overhead)
     OFF      All traffic is discarded
              (no practical use)
     ON       Non-IP incoming data is rejected
              at the IP level (default)
```

Can be useful in diagnosing performance problems (to detect discarded packets)

Some Informational Display Commands ...

Seldom Used Commands (cont'd)

Ù **SET RECORD=ON** (any load)

```
SET RECORD=ON         Logs task info on SYSLST, each time a
                      task terminates.
                      Default is OFF.
                      Full interpretation only by CSI.
```

```
IPNTXTCP Di 1628 Av  281 T 45770 Ac 890 Fx 0 Qu 0 Pr 7 Co 1 Pa 730

 - Di Dispatch count
 - Av Average usec this task was dispatched (Av=T/Di)

 - T  Accumulated Elapsed Time (ET) this task had control of
      the TCP/IP partition
      (is usually in VSE/ESA native cases similar to CPU-time)
      (allows to locate a CPU-dominating task)

 - Fx Number of scans of FiXed queue
 - Ac Number of scans of ACtive queue
 - Qu Number of QUick scans
 - Pr Number of scans of PeRsistence queue (hot spots of total)
 - Co Number of COmplete total scans
      (Should be as low as possible)
 - Pa Number of times control is PAssed directly from another task

 The statistics for 'long running TCP/IP tasks' is only contained
 in the shutdown.

 TELNET measurements (Service Pack J) showed an overhead for
 SET RECORD=ON of about 0.5% CPU-time
```

Ù **SET DIAGNOSE=PERFORM** (any TCP load)

```
SET DIAGNOSE=PERFORM   Provides additional statistics upon
                       termination of a connection, e.g. an
                       FTP session=transfer, or a logoff of
                       a Telnet user.
                       Very low CPU-time overhead.
                       Default is OFF.
```

```
For explanation of the IPT324I output lines, refer to next foil.
```

Some Informational Display Commands ...

Sample DIAGNOSE=PERFORM Output

```
 FTP Daemon Retrieving File, Count: 131070  Userid: SYSA

-------------( Performance Display )---------
 IP: 10.0.0.1 Port: 1030 Local Port: 20
                                                          In  Out
 Connection duration............     22519708  (usec)  X   X
   Maximum turn around time......       27185  ( "  )  X   X
   Transmission block count......         731          -   X
   Maximum depth count...........           1          X   -
   Maximum foreign window........       32768  (byte)  -   X
   Byte count of data sent.......    10747742          -   X
   Byte count of data resent.....           0          -   X
   Byte count of data received...           2          X   -
 SWS mode total time............     16160538  (usec)  -   X
   Number of times in mode.......         123          -   X
 Retransmission mode total time..           0  (usec)  -   X
   Number of times in mode.......           0          -   X
   Number of retransmissions.....           0          -   X
 Maximum Depth mode total time...           0  (usec)  X   -
   Number of times in mode.......           0          X   -
 Maximum window achieved.........        8112  (byte)  -   X
  Segments in window.............           2          -   X
```

```
- All times are in usec
- Inbound and Outbound data (TCP) are marked (In/Out) here on top!

- Connection duration   is the time from first to last byte
  (not including any setup/close time)
- Max. turn around time  is the max. time for an individual block
  from send time to ACK
- 'Transmission block' is a TCP Segment
- Byte count of data sent/received  is the total size of a file
  (in case of FTP)
- Max. Depth count  is the max. number of inbound packages enqueued
  in that single connection

- SWS  means 'Silly Window Syndrome'
  (Receiver gives a small window, filled very fast by the sender).

  Number of times in SWS mode  means how often window was closed
  and waits occurred until restart

- Retransmission mode  is entered as soon as the 6th TCP segment
  is being retransmitted

- Maximum Depth mode  means that TCP/IP is no more able to accept
  any incoming packets, until IBBLOCKs are freed
```

Some Informational Display Commands ...

Ù **DUMP ...**

```
Use this diagnostic command only when instructed.
```

TCP/IP 1.4 Commands

Ù **TRACERT** (any TCP load)

```
TRACERT any.domain.com   Allows to determine the path that
                         is being used to this domain
                         or IP address
```

```
This command is being provided as TCP/IP command,
but also execution as CICS transaction TRAC is possible
```

Ù **DISCOVER** (any TCP load)

```
 DISCOVER any.domain.com  Allows to determine the maximum
                          MTU size that should be used
                          for that connection

THE BEST MTU DISCOVERED: xxxxx
```

```
This command is being provided as TCP/IP command,
but also execution as CICS transaction DISC is possible
```

Remove Unnecessary Actions from TCP/IP

Remove Unnecessary Actions from TCP/IP

Ù **Symptom**

„ **TCP/IP partition consumes sporadically CPU-time, 'without doing anything'**

Ù **Background info**

„ **TCP/IP must inspect EVERY data packet it gets**

```
This includes also
     - Non-IP datagrams (e.g. Novell)
     - Mis-routed datagrams
     - ARP datagrams

QUERY STATS now inludes also counters for such traffic.
```

Ù **Recommendations**

„ **Make sure IP-Filtering is ON for OSA and 3172 etc**

```
TCP/IP for VSE/ESA should only see the datagrams directed
for itself
```

„ **Do not use an 'old gateway address' as IP address**

```
At least use 'SET GATEWAY OFF' to discard irrelevant data
earlier
```

„ **Find out the source for frequent ARP updates**

```
ARP requests from outside TCP/IP for VSE/ESA cannot be
avoided or influenced.

Use QUERY ARPS and check that the C: parameter is not high.
Naturally, SET ARP_TIME should not be too low, also.
```

Í **Filter away unnecessary data packets**

NFS Related Areas

NFS Related Areas

Buffers, Control Blocks, Modules

Area & Purpose	Size	GETVIS-	Note
NFS Modules (code)	160K 50K	-24 -31	a)
NFS Control Blocks	12K	-31	b)
NFS Directories	c)	-31	c)
File READ Caches (for file data)	d)	-31	d)
VSAM Attribute Tables (DIR cache)	e)	-31	e)
File WRITE Caches (for file data)	dynamic	-31	f)
File Request Blocks (FRBLOKs)	n x 4.5K	-24	g)
NFS Request Blocks	m x .16K	-31	h)

Notes:

a) Current size is about 160K. Prepared for RMODE=ANY.
Running in AMODE=31, except when SOCKET calls are done

b) About 8 to 16K, varying according to traffic

c) 1 NFS directory is built/required for each MOUNTed tree.
Each one is built by requesting GETVIS of DIRCACHESIZE byte (1 or multiple times) and by returning unused parts.
Total space required is very environment specific

d) Each File READ Cache is up to READCACHESIZE and kept at least for READCACHETIME seconds.

e) 1 VSAM Attribute Table is built/required for each VSAM MOUNT.
Its size is VSAMTABLESIZE, a fixed value

f) The size and number of File WRITE caches is dynamic

g) n is the number of FRBLOKs (1 per 'NFS session')

h) m is equal to MAXREQUESTS (default=1000), so 160K in total

NFS Related Areas ...

NFS Related Data Areas and Parameters

```
            -------
           |...    |      ------
          -------  |     |...   |
         |NFS    | |    -----   |
         |Direc- | | + |VSAM  | |
         |tories | |   |Attr. | |
         |       |-    |Tables|-
          -------       ------
       DIRCACHESIZE  VSAMTABLESIZE
----   (<80K each)    (48K each)    ---------- -----
    |((DIRGROUPSIZE))              |          |     |
    |  ((NFSTIMER))                |          |     |
    |      |                 A     |          |     |
    |<-----                  |     |          |     |
    |          ----------    |     | NFS      |VSE  |
UDP |         |...       |   .     | File I/O |I/O  |
and |        ----------  |   |Dir  | Routines |rou- |
IP  |       |File READ | |   .info |          |tines|
    |       |Caches    | |   |     |          |     |
    |       |          | |   .     |          |     |
    |<======|          | |<==========<==      |     |
    |       |          | |         |    |     |     |    ----
    |       |          |-          |    |     |     |   /      \
    |        ----------            |    |     |     |  |        |
    |       READCACHESIZE          |    |     |     |  |\      /|
    |       (<128K each)           |    |     |READ |  |  ----  |
    |     ((READCACHETIME))        |     =============<=|        |
    |                              |          |     |  | DASD   |
    |   A      ----------          |          |     |  | Files  |
    |Dir|     |...       |         |     =============>=|        |
    |updates ----------  |         |    |     |WRITE|    \      /
    |   |   |File WRITE| |         |    |     |     |      ----
    |   .   |Caches    | |         |    |     |     |
    |======>|          | |===========>==      |     |
    |       |          | |         |          |     |
----        |          | |          ---------- -----
            |          |-
             ----------
           # and size as req'd
          ((WRITECACHETIME))

- Sizes shown refer to default values
```

Some Perf. Related NFS Commands

Some Performance Related NFS Commands

Settings are usually included in NFSCFG.L (default member).
If omitted there, product defaults are used, which currently all are also contained in this shipped NFSCFG.L member.

The values can also be set from the console by prefixing 'NFS '.

You will be able to display all settings by NFS QUERY CONFIG.

Except indicated otherwise, NFS buffers may reside above the 16M line.
'NFS WRITE' means VSE inbound data, 'NFS READ' means outbound data.

All settings marked by *) are for internal use and should not be used, unless requested by Technical Support.

Ù DATAGRAMTRACE *) (all NFS loads)

```
DATAGRAMTRACE=YES|NO                         (default=NO)
```

When set, this allows to simply trace the UDP datagrams used for the transfer of NFS data.
YES will have some performance degradation, so only use when required. Default is NO.

Enter NFS DATAGRAMTRACE to see the current setting. (?)

Ù DEBUG *) (all NFS loads)

```
DEBUG=YES|NO                                 (default=NO)
```

When set, this trace simply shows the datagrams transferred.
Note that this trace is different from SET DEBUG.
DEBUG=YES will have a severe performance degradation, so in any case use the default (NO).

Enter NFS DEBUG to see the current setting.

Some Perf. Related NFS Commands ...

Some Performance Related NFS Commands (cont'd)

Ù DIRCACHESIZE (all NFS MOUNTs)

```
DIRCACHESIZE=nnnnK|M                         (default=80K)
```

NFS caches each NFS directory in 1 or more NFS directory blocks.
Each directory block is GETVISed (has an initial value) in blocks of DIRCACHESIZE bytes.
Each NFS directory is built at first reference, requesting as many directory blocks as required. When the entire directory is read, unused GETVIS space is being given back.

Each NFS directory exists until shutdown or until the directory is flushed.

Select your value

- such that most NFS directories fit into 1 directory block
  (1 entry is between 40 and 90 byte,
  a huge directory of about 6000 entries may need 512K)
- ample, since GETVIS-31 should not be a problem,
  and unused space is being returned

Since it is incremental, DIRCACHESIZE usually is not critical

Ù DIRGROUPSIZE (all NFS directory READs)

```
DIRGROUPSIZE=nnnn                  (default is 10 entries)
```

This value determines when directory data for a first 'DIR' command are being sent to the NFS client.
For subsequent DIRs, this parameter is ignored/not required.

A low value will give a fast first response, but may also increase the number of total IP packets or UDP segments sent.

A high number will avoid this 'clustering' of requests and may use less resources in total.

Some Perf. Related NFS Commands ...

Some Performance Related NFS Commands (cont'd)

Ù READCACHESIZE (all NFS READs)

```
READCACHESIZE=nnnnK|M                   (default is 128K)
```

```
All data of any file to be transferred to the NFS client is
being cached in a (file specific) File READ Cache.

Each cache is GETVISed once dynamically (READCACHESIZE) and
kept until the transfer of the file is completed, or until an
idle-time limit of READCACHETIME is expired.
If a file is < READCACHESIZE, the remainder is returned.
Each cache is being treated as 2 areas of equal size,
just to allow overlaps of emptying and filling it.

Since READCACHESIZE/2 usually is much bigger than any I/O
blocking, values bigger than its default are expected to be
only marginally better.
```

Ù READCACHETIME (all NFS READs)

```
READCACHETIME=nnnn                    (default is 30 sec)
```

```
Maximum time a File READ cache is being held w/o any actual
READ activity.
You may select a smaller value, but GETVIS-31 (where these
buffers are located) should not be a problem.

The selection of this value may also be impacted by the
amount of 'dirty READs', which NFS itself cannot avoid.
```

Ù MAXPACKETSIZE *) (all NFS WRITEs)

```
MAXPACKETSIZE=nnnnK|M                     (default is 8K)
```

```
This value determines the maximum size of the IP-data plus
header which is accepted by the NFS server when a file is
being written to VSE.

Helps to reduce the max IP datagram size coming in, provided
the NFS client is intelligent enough to adapt.

With Service Pack J, this parameter is ignored, since no more
required.
```

Some Perf. Related NFS Commands ...

Some Performance Related NFS Commands (cont'd)

Ù DEFINE FILE...NFSTIMER (all NFS dir. READs)

```
DEFINE FILE..NFSTIMER=nnn           (default is 0 sec)
```

```
Defines the time limit (in sec) for NFS to keep the file or
directory in its NFS directory cache. When expired, NFS causes
the cache to be rebuilt at the next file request.
The default of 0 means no automatic clearing of the NFS
directory cache for that file.

You may set NFSTIMER to a lower time value, if a file is
often used.
```

Ù VSAMTABLESIZE (all NFS VSAM WRITEs)

```
VSAMTABLESIZE=nnnnK|M                    (default is 64K)
```

```
This is the size of the VSAM attribute table (or DIRLIST cache),
where each accessible/MOUNTed file is listed.
About 80 byte is required for each entry/file.
When this table is too small, a VSAM file update will fail.
So, make it big enough, it resides in GETVIS-31 anyhow.
```

Ù WAKEUPTIME (all NFS loads)

```
WAKEUPTIME=nnnn                        (default is 5 sec)
```

```
This is the (unconditional) time interval after which certain
caches are are being inspected and potentially released, and/or
files closed.
It refers to: NFS Directories, File READ/WRITE Caches
```

Some Perf. Related NFS Commands ...

Some Performance Related NFS Commands (cont'd)

Ù WATCHDIRCACHE *) (all NFS MOUNTs)

```
WATCHDIRCACHE=ON|OFF                     (default is OFF)
```

```
This parameter traces the GETVIS allocation and deallocation
of the NFS directories, and thus may help in case of problems
with DIRCACHESIZE.
```

Ù WATCHREADCACHE *) (all NFS READs)

```
WATCHREADCACHE=ON|OFF                    (default is OFF)
```

```
This parameter allows you to watch the allocation of the File
READ Cache for a VSE file, and thus may help to properly
select READCACHESIZE.
```

Ù WATCHREADS *) (all NFS READs)

```
WATCHREADS=ON|OFF or YES|NO              (default is OFF)
```

```
This parameter allows you to trace all incoming READs.

Some overhead, use it only for debugging purposes
```

Ù WATCHWRITES *) (all NFS WRITEs)

```
WATCHWRITES=ON|OFF or YES|NO             (default is OFF)
```

```
This parameter allows you to trace all incoming WRITEs.

Some overhead, use it only for debugging purposes
```

Some Perf. Related NFS Commands ...

Some Performance Related NFS Commands (cont'd)

Ù WATCHWRITECACHE *) (all NFS WRITEs)

```
WATCHWRITECACHE=ON|OFF                   (default is OFF)
```

```
This parameter allows you to watch the allocation of the
WRITE Cache for incoming data.

Some overhead, use it only for debugging purposes
```

Ù WRITECACHETIME (all NFS WRITEs)

```
WRITECACHETIME=nnnn                   (default is 30 sec)
```

```
This is the time interval after which the File WRITE Cache
for an incoming file request is being released, provided
no activity took place during this interval since the last
record arrived.

It also causes the file being closed then, if required,
and thus remaining data are flushed out.

You may set this value to 15 sec, in case you may have a
temporary GETVIS-31 problem and cannot bring TCP/IP down
```

Ù MAXREQUESTS (all NFS loads)

```
MAXREQUESTS=nnnn                        (default is 1000)
```

```
Every request from a client needs an NFS request block of about
160 byte, as long as this request is still 'in use'; e.g. for
a DIRLIST request a long time, for a READ a much shorter time.

The default of 1000 should be sufficient for most cases.
A too low value will not allow the additional concurrent
function, but displays the current NFS request block usage.
```

TCP/IP for VSE/ESA Performance PTFs

TCP/IP for VSE/ESA Performance PTFs

Note that each TCP/IP 1.3 PTF is a full replacement and thus does not have any TCP/IP pre-req (they get bigger and bigger).

IBM APAR	IBM PTF(s)	Subject	'CSI ServPack' Date
PQ11216	UQ12233	Miscellaneous	2.3.0-GA 97-12-09

This PTF provides few missing modules, new functions, and gives some reduction in CPU-time for TN3270 daemons and other TCP/IP loads, by a streamlined internal task structure.
(This PTF together with VSE/ESA 2.3.0 as of 97-12-05, here is referred to as '2.3.0 GA-level')

PQ11981 PQ12876	UQ13349 UQ14494	Performance, superseded	98-01-20 'E' 98-02-15

This PTF measurably reduces the required CPU-time for TCP/IP loads (especially for higher number of Telnet users).
Most of the reductions stem from a more efficient setup of TCP/IP internal queues and search algorithms.

PQ14724	UQ16971	Performance for FTP	'F' 98-04-xx

This PTF reduces CPU-time requirements for FTP by better internal buffering (VSAM, POWER, and SAM variable records).
TCP/IP dispatching in general was enhanced.

PQ14716	UQ19196	NFS function, etc.	'G' 98-07-03

This PTF introduces NFS and 2216 support, announced 98-05-07.
It also comprises performance enhancements
- Restructured TCP/IP subtasks
- Reduced Non-Parallel Shares
- FTP kernel changes (incl. MAX_BUFFERS enhancements)
- Improved SET DIAGNOSE=PERFORM displays
- New QUERY ARPS display
- API Socket appl's now also 31-bit mode

PQ18295	UQ20719	Misc, etc.	'H' 98-08-31

This PTF introduces misc. enhancements, e.g. AUTO-FTP, incl. slight performance enhancements for FTP and LPR.

TCP/IP for VSE/ESA Performance PTFs ...

TCP/IP for VSE/ESA Performance PTFs (cont'd)

PQ19496	UQ22503	Misc, etc.	'I' 98-10-30

This PTF includes misc. modifications/enhancements and the SET FIXED_RETRANS setting.

PQ20942	UQ26288	VSCR, etc.	'J' 99-02-12

This PTF, again, introduces misc. functional changes and performance enhancements, like
- VSCR by moving several control blocks and buffers above the line (most benefit for Telnet)
- FTPBATCH program, running in a separate partition
- Reduced CPU-time for interfacing with POWER files (FTP, NFS)

PQ24008	UQ30758	Misc. etc.	'K' 99-06-11

This PTF, again, introduces a huge number of modifications and enhancements, like
- optional new command processor

plus functional enhancements for
- FTPBATCH
- display
- trouble shoot (DEFINE TRACE for all incoming TCP traffic)

New dispatch scheme with 'fast dispatches'

PQ27233	UQ32439	GPS	99-07-17

General Print Server as a priced feature

PQ27252	UQ38659	Misc. etc.	'L' 99-11-xx

This PTF, again, introduces a big number of modifications and enhancements, like
- SET DIAGNOSE=SWS
- improved directory access for FTP
- some detection of orphaned storage

Be aware of problems when installing a CSI Service Pack on top of the TCP/IP for VSE/ESA IBM shipped product.
As indicated, this is an unsupported environment.

TCP/IP for VSE/ESA 1.4

TCP/IP for VSE/ESA 1.4

PQ29053	UQ44071	Rel 1.4	2000-06-15

New Release TCP/IP for VSE/ESA 1.4

Mostly functional enhancements, plus
- Implementation of Slow Start mechanism (outbound TCP)
- CHECKSUM calculation also in H/W
- ...

PQ40278	UQ48729(2.3/2.4)
	UQ48724(2.5)
		Misc	'A' 2000-11-14

Many fixes and slight enhancements
(refer to PQ40278 + II12618), plus
- Better algorithms to reduce re-transmissions
- TRACERT and DISCOVER commands
- ...

PQ45314	UQ55343(2.3/2.4)
	UQ55344(2.5)
		Misc	'B' 2001-06-25

(SSL/TLS support not possible in IBM TCP/IP 'B')
Multi event processing
- more than 1 'auto-event' can be scheduled and can run concurrently (AutoLPR, AutoFTP)

PQ.....	UQ.....	TBD	'C' 2001-xx-xx

TBD

TCP4VSE Performance Results - General

PART E.

TCP4VSE Performance Results - General

Ù General Aspects

Ù Measurement Environments & Tools

General Performance Issues

General Performance Issues

Usual types of performance data:

Ù **Resource Consumption of an Activity**

- CPU-time, #I/Os, storage ...

required to perform a certain TCP/IP activity

```
(e.g. - to use TELNET for CICS transactions,
 or   - to transfer 1M of data)
```

Ù **Achieveable Performance Values**

(Response/Elapsed times, Data Rates, Thruput)

For example, ...
What response times to expect at a certain TN3270 transaction rate?
What effective data rate (EDR) can I achieve for 1 single FTP activity in my environment?

plus impact on other loads

Ù **Resulting Usage of System Resources**

- CPU utilization, I/O rate, Storage

at a certain activity level

Also required for Setup and Capacity Planning purposes.

General Performance Issues ...

General Performance Issues (cont'd)

Ù **Impact of Performance Parameters**

What effect has, based on the current situation, a specific performance relevant parameter change?

„ **If possible, change only 1 parameter at a time**

„ **Parameter sensitivity varies**

It may well be that changing a parameter in your environment may not produce any delta, since another resource represents a bigger bottleneck.

BUT, after having changed the biggest bottleneck, the same change may have an impact you directly can see.

This is especially understandable e.g. for FTP, where all components in the chain must work with the same global speed.

Even if e.g. the network is capable of much higher overall data rates, the throughput will be limited/synchronized by e.g. the average speed of the DASDs.

„ **Check before change**

Before changing a parameter, make sure that this parameter can have at all an influence on the type of workload(s) you consider.
To that end, refer e.g. to a previous foil 'TCP/IP Performance Relevant Parameters'

Ù **What are performance-optimal values?**

- The optimal selection of performance-relevant setup or operational parameters is often very important.

Measurement Setups

Measurement Setups

```
 A) Connection to S/390 Host
 ---------------------------

           Driver system                System under Test (SUT)
          ---------------              ----------------
         |  9221-421     |            |  9672-Rx1'     |
         | VM/ESA 1.2.2  |   Real CTC |  VSE/ESA 2.3   |
         | TPNS 3.5      |=================|             |
         | TCP/IP-VM 2.4 |   (ESCON)  | TCP/IP for VSE |
          ---------------              ----------------
             |                            |
             |                            |- H/W Monitor
             |- RAMAC Array Subs-2        |- 'Old' 9345s
             |- Virt. Disk                |- Virt Disk
```

This configuration is similar to our traditional setup for non-TCP/IP online workloads (VTAM SNA) with has/had a parallel channel each to a 3745 with NCP. This configuration is used for TN3270, and so far for all FTP applications (LIBR, POWER, VSAM ESDS).

```
 B) Connection to RS/6000
 ------------------------
                                        System under Test (SUT)
          ---------------              ----------------
         |   RS/6000    C|            |  9672-Rx1'     |
         |   Model 570  L|   Real CTC |  VSE/ESA 2.3   |
         |              A|=================|           |
         |              W|   (ESCON)  | TCP/IP for VSE |
          ---------------              ----------------
              |                          |
              |                          |- H/W Monitor
            - - - -                      |
  -----    | Token |                     |-'Old' 9345s
 |PS/2 |--  Ring                         |- Virt. Disk
 |     |   |16mbps |
  -----     - - - -
```

So far, this RS/6000 configuration was used for FTP with VSAM ESDS.

Our primary task here was seen
- to optimize the TCP/IP for VSE product itself
- to provide optimal guidelines for it.

Network and Communication performance results for TCP/IP have been published widely and are not VSE/ESA specific.

Regarding the 'old' types of disks, only for the measurements with FTP the disk speeds were of influence to TCP/IP itself.
To assess faster disks, also virtual disks were used in some cases to show DASD speed impact (and to extrapolate to 'todays real disks').

Measurement Scheme

Measurement Scheme (TELNET)

The following sequence was applied to each TELNET measurement run.
It assumes a single TCP/IP partition in Fy.

Production run:

```
 All parameters correctly set for startup, VTAM started in F3,
 CICS(es) also started and ready (we use F4/F5)

 * Start TCP/IP partition (Fy)
                                        ====>  'TCP/IP up'
   SIR, PRTY, VOLUME
  (SET DEBUG=FULL)  (reset later, use with care)
   DEBUG (VSE, must be OFF)
   MAP, MAP REAL, MAP Fy
   GETVIS SVA, GETVIS F3, GETVIS Fy
   D NET,BFRUSE

 * Enable TELNET logon to outside VSE
                                        ====> 'Sessions up' (ACT/S)
   MAP REAL
   GETVIS SVA, GETVIS F3, GETVIS Fy  ... to check TCP/IP TN3270 Logon
   QUERY STATS and QUERY ISTATS      ... to get info BEFORE any real
                                         TN3270 traffic starts
 * Enable TELNET activity
                                        ====> 'Traffic up'
  (After total activity is stable, just before measurement start)

   QUERY VERSIONS
  (QUERY ALL)
   QUERY SET
   QUERY STATS and QUERY ISTATS, QUERY TRACES
   SET DEBUG=OFF

   MAP REAL, MAP Fy
   GETVIS SVA, GETVIS F3, GETVIS Fy
   D NET,BFRUSE  + DSA display

 * At measurement interval begin (-2sec):

   'TPNS'+'VM'
   QUERY STATS and QUERY ISTATS
   SYSDEF TD,RESETCNT
   ////////////// MEASUREMENT (10 min) //////////////
   QUERY TD,INTERNAL
   QUERY STATS and QUERY ISTATS
   'TPNS'+'VM'
  (QUERY ALL)

   D NET,BFRUSE  + DSA display
   GETVIS SVA, GETVIS F3, GETVIS Fy
   MAP REAL, MAP Fy
```

Measurement Scheme ...

Optional Full Monitoring run:

```
  * At post measurement interval begin (-2sec):

    QUERY STATS and QUERY ISTATS

 -> SIR MON=ON
    'TPNS'+'VM'
    SYSDEF TD,RESETCNT

    ////////// 'POST MEASUREMENT' (10 min) ///////////

    QUERY TD,INTERNAL
 -> SIR MON                  ...to display SVC, FC and BOUND stats

    QUERY STATS and QUERY ISTATS
    'TPNS'+'VM'

    D NET,BFRUSE
    GETVIS SVA, GETVIS F3, GETVIS Fy
    MAP REAL
 -> SIR MON=OFF


'VM' means       NETSTAT POOLSIZE
                         GATE
                         ALL

Also of interest:  Total CPU-time for TCP/IP for VM
                   TCP/IP for VM profile
```

Tools Used

- VSE/ESA Display System Activity in IUI
- QUERY TD statistics
- TPNS (Teleprocessing Network Simulator) under VM/ESA. Used for TN3270
- Hardware monitor for 9672 CMOS processor. The processor itself is a 1 to 6-way, but -for internal reasons- running at lower speed than a 9672-Rx1

TCP4VSE Performance Results - TN3270

PART F.

TCP4VSE Performance Results - TN3270

Ù TN3270 Results & Hints

„ CPU-time Overhead and Requirements

„ Virtual Storage Capacity

TN3270 Performance Results

TN3270 Measurement Results for DSW/LE

Environment

- VSE/ESA 2.3.0/2.3.2 + TCP/IP 1.3 Status 01/98 (SPack E), 07/98 (SPack G), 03/99 (SPack J), 09/2000 (1.4)
- DSW online workload, set up with COBOL/LE
- VTAM 4.2 (F3)
- 2 CICS/VSE partitions (F4,F5)
- TCP/IP for VSE/ESA (F7)
- F4 and F5 partition balanced with F7, by default

Measurement runs

- 125 active terminals per CICS partition, driven by TPNS ('2x125'). Different loads created by different #terminals or thinktime. Default thinktime TT was 11 sec.
- Each run lasted 10 minutes, after a stabilization interval.
- TD was used by default, but also SD.
- POOL=YES was used with TELNETD_BUFFERS=20, but also POOL=NO runs were done
- Except indicated otherwise, runs were done with a single engine.
- ALL terminals were 'converted' from VTAM SNA to TCP/IP

TN3270 Performance Results ...

TN3270 Measurement Results for DSW/LE

```
TCP/IP Serv.Pack E, runs done 98-01-13   (E)
  "    Serv.Pack G, runs done 98-07-17   (G)
  "    Serv.Pack J, runs done 99-03-05   (J)
  "    Serv.Pack K, runs done 99-07-07   (K)
  "    1.4        , runs done 2000-09-14 (4)
```

Run	Case	Var.	CPU ut. %	txn /sec	Avg. RT sec	CPUT: msec /txn	NPS	Delta: msec /txn	ITRR
Runs with VTAM SNA (no TCP/IP)									
SV1	SD 2x200		50%	26.63	0.19	18.96	-	Base2	1.00
SV2	SD 2x125		30%	16.70	0.21	18.27	-	-	-
TV1	TD 2x200		54%	26.63	0.23	20.81	0.289	Base1	1.00
TV2	TD 2x125		34%	16.73	0.22	21.02	0.300	-	-
Runs with POOL=NO									
TN1	TD 2x125	E	70%	18.39	0.39	42.33	0.623	21.52	0.45
TN1	TD 2x125	G	71%	16.74	0.25	43.44	0.365	22.63	0.44
TN1	TD 2x125	J	70%	16.69	0.23	42.97	0.388	22.16	0.44
TN2	TD 2x125	HW 4	69%	16.80	0.21	41.41	0.332	20.60	0.50
Runs with POOL=YES									
TY2	TD 2x 50	E	28%	6.72	0.33	44.65	0.639	23.84	0.465
TY3	TD 2x125	TT5 E	95%	23.57	0.97	40.82	0.612	20.01	0.51
TY4	TD 2x125	TbC E	71%	16.68	0.38	43.50	0.627	22.69	0.48
TY5	TD 1x125	K	40%	8.40	0.19	48.52	0.380	-	-
TY1	TD 2x125	E	70%	16.70	0.28	43.86	0.633	23.05	0.47
TY1	TD 2x125	G	76%	16.64	0.24	46.41	0.390	25.60	0.45
TY1	TD 2x125	J	74%	16.69	0.24	44.84	0.425	24.03	0.46
TY1	TD 2x125	K	74%	16.62	0.24	45.63	0.419	24.82	0.46
TY1	TD 2x125	4	70%	16.72	0.23	42.31	0.346	21.50	0.49
TY6	TD 2x125	HW 4	70%	16.81	0.25	42.07	0.344	21.26	0.495
SY1	SD 2x125	E	64%	16.68	0.22	43.86	-	24.90	0.46

```
TbC    TCP/IP priority below CICS priority
TT     Thinktime in sec (default is 11 sec, TT5 =5 sec)

- Results for SPack G:  Slightly increased CPU-time,
                        but significantly lower NPS for TD
- Results for SPack J:  CPU-time back on E, plus huge VSCR
- Results for 1.4    :  7% to 13% less TCP/IP CPU-time ovhd
```

TN3270 Performance Results ...

Some TN3270 Measurement Observations

Ù Important Note

When working with TCP/IP and when analysing the results, it became more and more clear that ....

Measurement setup for TN3270 is some kind of worst case here

... regarding TCP/IP CPU-time overhead.

This is caused
- by the fact that only 1 port (source) and 1 port (target) is used,

combined with the internal design of TCP/IP for VSE/ESA:

The more 'packets' TCP/IP finds when visiting a level of the TCP/IP stack, the more effective can it work.
Traces have shown that in 9x% of all cases at most a single 'packet' was eligible for being promoted up or down the stack.

Due to visible potential for performance improvements, currently highest priority was given to product improvements, rather than investigations on representativeness.

Ù VTAM Base Measurements

- Only small dependency of CPUT/txn from CPU utilization. Varies between about 19 to 21 msec
- TD vs SD overhead depends on the CPU utilization:

  15% for low, 10% for medium, and 5% for high utilization

Ù TCP/IP Measurements with Variations

- Total CPUT/txn varies between about 41 to 45 msec

TN3270 Performance Results ...

Some TN3270 Measurement Observations (cont'd)

Ù TCP/IP Overhead in terms of CPU-time/txn

```
Delta/txn = msec/txn(TCP/IP) - msec/txn(VTAM)
```

„ Varies here between 20.0 and 24.9 msec

- Overhead is lower for high traffic, higher for low traffic

  This effect is to be considered for capacity planning

- POOL=YES overhead is about 5 to 7% higher than for POOL=NO

  This is the cost for sharing TN3270 buffers, allowing a higher TCP/IP partition capacity for Telnet, before Service Pack J.

- Overhead with SD here is not less than for TD
- TCP/IP overhead is lower when TCP/IP has lower priority

  The small response time overhead when giving TCP/IP a lower priority suggests to give TCP/IP not highest priority in order to save some CPU cycles

„ Overheads for Service Pack E are 20% and more lower than for 'TCP/IP 2.3.0 GA'

„ Service Pack J overheads regained the values of Service Pack G

TN3270 Performance Results ...

Some TN3270 Measurement Observations (cont'd)

Ù Assessment of TCP/IP Overhead

```
                      msec/txn (VTAM)
ITRR = ITR ratio = ----------------
                     msec/txn (TCP/IP)
```

This ratio (and thus the relative TCP/IP overhead) depends directly on the total CPU-time (or pathlength) of a customer's average transaction.

To be independent of processor speed, and for simplicity reasons, let's turn over to (approximate) pathlengths and 'MIPS'.

In the measured cases, average overall (VTAM based) CPU-time of a transaction was about 20 msec, corresponding to about 280KI.
TCP/IP overhead was between 280KI and 350KI.

Average customer transaction pathlength may vary between say 300K ('300 KI') and 1 Million instructions:

```
Average todays txn-pathlength:

         Total-CPU-time by online txns x 'MIPS'
         --------------------------------------
                 #txns in that interval
```

Some TN3270 Measurement Conclusions

Expected rel.CPU-time and ITR-ratio vs SNA		
Type/CPU-Heaviness of Load	Rel. CPUT w/ TCP/IP	ITRR
DSW, measured 280KI	2.0	0.5
Medium cust.txn 560KI	1.5	0.67
Heavier cust.txn 840KI	1.33	0.75
Heavy cust.txn 1000KI	1.28	0.78
Your workload .. ____KI	_.__	0.__
- ITR-ratios and total overhead only apply to TCP/IP related terminal activity.		

Response time impact is small: about 0.17 sec delta, here.

This delta is the same, independent of the pathlength of a txn.

TN3270 Processor Capacity Planning

Processor Capacity Planning Examples

Ù VSE/ESA Native

VSE/ESA on a 2003-207 processor (about 24 MIPS).
50% CPU utiliz. during peak hour, 20 txn/sec,
plus 20% batch ==> 70% total CPU

Part of the terminals are now being attached via TCP/IP, here a subset causing 10 txn/sec (with the same mix).

0.50x24MIPS / 20 txn/sec = 600 KI/txn avg. txn-pathlength

Calculation:

The additional CPU power required is:

10 txn/sec x 280 KI/tx = 2800 KI/sec TCP/IP overhead = 2.8 MIPS

2.8 MIPS is about 2.8/24= 12% CPU utilization.

So Online work increases from 50% to 62%. So still enough CPU power is available for concurrent batch.
Total CPU utilization will be 82% (at same throughput).

Online utilization increases from 50% to 62%

Ù VM/VSE Guest

For simplicity, the same situation (VSE throughput and CPU utilizations) is assumed here as in the native case above.

VM/VSE guest with a T/V ratio of 1.20.

Total utilizations (including VM/CP overhead related to the guest):
50% x 1.2 =60% Online related
20% x 1.2 =24% Batch ==> 84% total CPU

Calculation:

Overhead is about 2.8 MIPS x1.2 = 3.36 MIPS = 14% CPU, and starts to impact batch throughput.

Online utilization increases from 60% to 74%

Telnet Capacity of a TCP/IP Partition

TN3270 Partition Capacity (before Serv. Pack J)

Before Service Pack J, Telent capacity of a TCP/IP partition was limited essentially by the amount of virtual storage below the 16M line.

Ù Virtual Storage (-24) Consideration for TELNET

P = PA + GD + GN + GS + GR

```
P   is the actual total partition size below the 16M line.
    It can be easily made equal to the private space below
    the 16M line, which is  10, 11, or 12 MB.

PA  is the Program Area size.  It is recommended to use
    SIZE=IPNET, which gives (independent of NFS)
    744K/768K/788K/888K for Service Pack G/I/J/K.

GD  is the required GETVIS-24 for defined TN3270 daemons.
    We observed 14.4K/13.0K per defined POOL=NO/YES daemon

GN  is the required GETVIS-24 for nonshared (POOL=NO)
    daemons and is 2x8K each (allocated when active)

GS  is the required GETVIS-24 for shared (POOL=YES) TELNET
    daemons and is #TELNETD_BUFFERS x 2 x 8K (allocated
    when SET), plus about 1K per daemon (measured)

GR  is the amount of GETVIS-24 you may/must reserve, e.g.

      - 1008K for being in MSG Fx mode (command processor,
        currently required for parsing input commands)
      - enough GETVIS for TRACEing
      - some area for NFS code and FRBLOKs
    Here, no NFS is considered, since most of it resides
    above the line.
```

Using the constant values (PA and GR w/o NFS), it results

```
   P - GA - GR   =    8620K   for P=10M private space
                   or 10668K   -"- P=12M      -"-

                 = GD + GN + GS
```

Telnet Capacity of a TCP/IP Partition ...

Ù Calculation Examples (before Service Pack J)

a) POOL=NO TN daemons only (GS=0)

```
                                  GD + GN
 Number of POOL=NO daemons =   -------------    = 185 (10M)
                               (14.4 + 16)K     = 230 (12M)
```

b) POOL=YES TN daemons only (GN=0)

```
- Assumption A (confirmed by measurements):

    It is on the safe side to define

            1.5 x tx_rate     TELNETD buffers

      tx_rate is the tx-rate in tx/sec

      i.e. 15 TELNETD buffers for 10 txn/sec.

      Or   1.5 x #act.termnls x (#txn per termnl and minute)/60

      With 1.0 to 3.0 txn per terminal and minute...
        1.5 to 4.5 x (#act.termnls/60)  TELNETD buffers
      i.e. 2.5 to 7.5 TELNETD buffers for 100 terminals

- Assumption B (to be adapted to specific environments):

        Each TN terminal produces about 2 txn/min

      i.e.  1 tx/sec corresponds to 30 terminals
       or   100 terminals produce 3.33 txn/sec
            and thus need 1.5 x 3.33 = 5 TELNETD buffers

- Calculation:

      The GD + GS required delta for e.g. 100 terminals:
       100 x 13.0K + 3.33 txn/sec x 1.5 x 2 x 8K + 100K
       = 1300K + 80K + 100K = 1480K
      Dividing the available GD+GS size by 1480K gives
```

```
   8620/1480 = 5.82  => 580  (POOL=YES) avg intensive
  10668/1480 = 7.21  => 720  TELNET terminals/sessions
```

Old Cmd Processor

Space Relief for Cmd Processor

Before Service Pack K ...

if you really have problems with space below the line ...

Í You may run the command processor from another VSE partition, using the TCP/IP Batch Facility

Not elegant, BUT very effective

Ù Command Processor Space Circumvention

```
R RDR,PAUSExy

// EXEC IPNETCMD,IZE-IPNETCMD,PARM='ID=nn'

 Enter TCP/IP console commands for the TCP/IP partition
 with ID=nn, as specified above (default is 00).

 The command processor is not required for the TCP/IP
 partition
 (except you would also enter console commands from there).

  - Use xy as REPLID.
  - Let the xy REPLID stand forever.
```

IPNETCMD uses XPCC under the cover to route TCP/IP commands to the selected target TCP/IP partition.

New Command Processor in Sevice Pack K

Refer to the following chart

New Command Processor

New Command Processor (Serv. Pack K)

Ù Now written in Assembler

„ Much faster than before

„ Much smaller size

Now moved fully into phase IPNET in 24-bit.

	Serv.Pack J	Serv.Pack K
IPNET size	788K	888K

Going into reply mode via MSG F7 does no more cost

- GETVIS-24
- heavy CPU consumption

Startup is automatically adjusted, if SIZE=IPNET is used

Ù Old command processor still selectable in 1.3 via OLDPARS initialization parameter

TN3270 Partition Capacity (Serv. Pack J)

TN3270 Virtual Storage (Serv. Pack J)

Ù **GETVIS Used Results for Telnet**

Measurement setup as described in chart for DSW/LE results

	POOL=YES -24 GETVIS	 -31	POOL=NO -24 GETVIS	 -31
TCP/IP just started	2752K	2124K	2752K	2128K
All sessions started	2760K	2132K	2760K	6168K *2
Telnet fully active	2760K	2208K *1	2760K	6268K

- 250 Telnet daemons, SET TELNETD_BUFFERS=20 for POOL=YES
- Partition size 40M, 12M below incl. 788K EXEC size
- Time instant of allocation of Telnet buffers:
  POOL=YES: when required *1 (5 to 20 were used)
  POOL=NO : at session logon *2
- At all times, the (old) cmd processor was loaded (below)

„ **No GETVIS-24 delta anymore between POOL=YES and NO**

Thus same total capacity for both POOL definitions, regarding storage below the line.

„ **POOL=NO needs more space (vs POOL=YES), but in TCP/IP GETVIS-31**

The delta here (4060K) is from 250x2x8K =4000K Telnet buffers (minus up_to_20x2x8 = up to 320K for POOL=YES buffers).

í **You may use POOL=NO and save about 5% CPU-time at cost of GETVIS-31 storage. Recommended**

í **By moving Telnet buffers above the line Telnet capacity has increased a lot**

For details refer to Service Pack K capacity

VTAM Startup Variations for Telnet

VTAM Startup Variations for Telnet Capacity Check

These VS measurements were done with Serv. Pack K,
but the conclusions apply to other service packs as well.

Naturally, here enough space above the line was available.

Also the VTAM APPL B-book for TCP/IP was varied.

Ù **Conclusion**

By careful setup of the VTAM APPL B-book, you may save some space, but only above the line

- No deltas seen in any space used below the line
  (neither in GETVIS nor in SIZE)
  -> no VS bottleneck seen here

- Negligible deltas above the line BEFORE TCP/IP start

- Deltas seen ABOVE the line, after TCP/IP start:

	400 termnls EAS=1	400 termnls EAS=dflt	200 termnls EAS=dflt
TCP/IP GETVIS	Same	Base	-32K
VTAM GETVIS	Same	Base	-64K
SVA	-972K	Base	-608K

- EAS is the Estimated Active number of Sessions, this 'application program' (Daemon) will have with other LUs. Is always 1 for each Telnet terminal. The default is 509 (or 256?)

í **EAS=1 saves about 2.4K/terminal in SVA-31**

í **1 terminal costs about 0.15K in TCP/IP, 0.3K in VTAM, and 3.0K (EAS=dflt) in SVA-31**

í **Telnet capacity not limited by VTAM books**

Telnet VS-Capacity (Serv. Pack K)

TN3270 VS-Capacity (Serv. Pack K)

Ù **GETVIS Used Results for Telnet**

Measurement setup as described in chart for DSW/LE results.

In all cases, POOL=YES was used (99-07-07).

GETVIS USED	TCP/IP -24	 -31	SVA -24	 -31	VTAM -24	 -31
a) 250 Telnet daemons, SET TELNETD_BUFFERS=20						
Only POWER started	-		436K	520K	-	
Only VTAM started	-		1036K	3340K	164K	4240K
TCP/IP just started with old cmd proc	1708K (2716K)	2184K -"-	1120K	4600K	164K	4396K
TCP/IP just started (new cmd proc)	1584K	2188K	1080K	4512K	164K	4396K
All sessions started	1592K	2196K	1112K	4796K	-"-	4504K
Telnet fully active	-"-	2288K	1120K	-"-	-"-	-"-
b) 125 Telnet daemons, SET TELNETD_BUFFERS=20						
TCP/IP just started	1108K	1588K	1116K	3988K	164K	4344K
All sessions started	1116K	1600K	1100K	4084K	-"-	4392K
Telnet fully active	-"-	1668K	-"-	-"-	-"-	-"-
c) 125 Telnet daemons, SET TELNETD_BUFFERS=30						
Telnet fully act.*1	1116K	1836K	1100K	4084K	164K	4392K

- TCP/IP partition was 40M, 12M below incl. 888K EXEC size
- VTAM partition was 20M
- New cmd processor was used, except in 1 variation of a)

*1 Same values w/ 30 buffers, except TCP/IP GETVIS-31

- All VTAM startups used a B-book for APPLs with 400 terminals and with EAS default (256 or 509 ?)

Conclusions are given on the next charts.

Telnet VS-Capacity (Serv. Pack K) ...

TN3270 VS-Capacity (cont'd)

Observations:

í **New cmd processor saves about 1.1M**

In TCP/IP GETVIS-24, in reply-mode.

í **Message Traffic costs below the line is negligible**

í **Cost of Telnet Daemons**

	125 daemons -24	 -31	Per daemon -24	 -31
TCP/IP GETVIS	476K	600K	3.8K *)	4.8K
VTAM GETVIS	0K	52K	0K	0.4K
SVA	20K	524K	0.16K	4.2K

Remember from other variations:

> Different setup of VTAM APPL B-book for Telnet did not show any deltas below the line

Conclusion:

í **Rough estimate for TN3270 VS-Capacity**

Based on *) above

```
Max. #TN daemons = (remaining GETVIS-24) / 4K
```

'Remaining GETVIS-24' is that GETVIS which is available in your TCP/IP partition, IF you would start it again, BUT w/o any Telnet daemons defined.

Here, the remaining GETVIS-24 was about 10M, resulting in (theoretically) 2500 TN daemons.

Much higher Telnet capacity per TCP/IP partition

TCP4VSE Performance Results - FTP

PART G.

TCP4VSE Performance Results - FTP

Ù FTP General Hints

Ù FTP Results

„ Interactive FTP with various file types

„ Interactive vs Batch FTP vs FTPBATCH

Ù Resource Planning

General FTP Related Aspects

General FTP Related Aspects

EDR = Effective Data Rate (KB/sec)

Usually it denotes the rate for a single FTP transfer,
but for capacity planning the aggregate EDR must be used.

„ **Achievable EDRs w/o TCP/IP**

IND$FILE (Workstation File Transfer):

Based on customer experiences and statements, about 30 to 50 KB/sec can be achieved as (single thread) EDR.

LANRES/VSE:

Higher EDRs can be seen (100 to 200 KB/sec, sometimes 300 KB/sec), also depending on the parameters cited above.

(Let us know if your experience should differ significantly)

„ **It is irrelevant, who initiated an FTP transfer**

There is the same EDR whether the FTP of a file from A to B was initiated via GET in system B or PUT in System A.

„ **To transfer a file from A to B may differ in EDR from transferring the identical file from B to A**

Even if identical FTP parameters and local file definitions are used, differences in effective DASD speeds may come into play:

- speed of the physical HDD
- type of READ and of WRITE caching
- blocksize used (KB/IO)

„ **The higher the EDR of an FTP transfer, the higher is the required CPU utilization**

Some key value for a given FTP setup is e.g.

'MIPS consumed per 100 KB/sec'

or

'KI consumed per KB transferred' ('KI per KB')

Effective FTP Data Rates

Effective FTP Data Rates

„ **Achievable Effective Data Rates (EDRs) depend on both ends and the connecting network**

```
     TCP/IP partner                              VSE host
         -------------                   ----------------
  ----  |      |      |     ---------     |        |       |
----
 /    \ |  FTP  TCP/IP| EDR| TCP/IP  |EDR | TCP/IP    FTP  | /
\
 \    /=| appl.|      |<==>| Network |<==>| for VSE| appl. |=\
/
  ----  |             |    |         |    |                |
----
  Disk  |      |      |     ---------     |TCP/IP partition|
Disk
         -------------                   ----------------

    Source or Target                          Target or Source

      Actual file transfer is from Source to Target,
      independent who initiated the FTP as a client.
```

Relevant Parameters for ...	FTP Speed(s)			CPUT/KB
	Source	Target	Network	
Network speed and load	-		X	-
TCP/IP parameters	X	X	x	x
FTP parameters	X	X	x	x
DASD speed (READ/WRITE)	X	X	-	-
Local file definition *1			-	
- type	X	X	-	X
- log. record length (NFS)	x		-	
- blocksize on disk	x	x	-	x
- I/O blocking (KB/IO)	X	X	-	x
- ASCII/EBCDIC/BINARY	x	x	-	X
Size of file(s)	X		-	x
Processor speed	X	X	-	X
Other concurrent activities	X	X	x	-
TCP4VSE PTF level	X	X	x	X

X Yes, parameter is relevant
x Parameter with smaller impact
*1 Especially important on VSE side(s)

The table above mainly holds for SINGLE FTP transfers.

Effective FTP Data Rates

Maximum aggregate EDRs for MULTIPLE FTP transfers depend on the utilization of all involved resources

Effective FTP Data Rates ...

Effective FTP Data Rates (cont'd)

„ EDRs displayed by TCP/IP for VSE

Starting with TCP/IP level UQ16971 (Service Pack F), the following data are being displayed to the initiator of the FTP transfer:

```
Transfer sec: 20.93   (524 KB/sec)
File I/O sec: 11.76   (954 KB/sec)
```

```
  | Start      X-fer of all TCP segments    End |
  V---------------------------------------------V

/===/  /=====/ /===/ /==/   /======/ File I/O routine
```

```
Transfer sec:  Elapsed time from the start of the transfer of
               the first TCP segment till the last TCP segment
               was ACKnowledged (includes idle times)

            -> Actual EDR w/o setup overhead, but with internal
               delays

File I/O sec:  The sum of all times the TCP/IP File Routine
               needed to complete logical file requests

            -> This would have been the resulting EDR,
               in case no TCP transfer (only File-I/O)
               would have been done
```

If the File I/O rate is very close to the transfer rate, ...

- the disk (including type of file access) is presumably the area which determines the overall FTP data rate. Check file definition/IO settings. To confirm, you may try a VSE Virtual Disk; or use the $null 10M file, created by FTP in VSE virtual storage.

If the File I/O rate is much higher than the transfer rate, ...

- the disk (including type of file access) does not represent a bottleneck. Instead, the TCP network transfer may be the determining factor.

In addition, you also may check the CPU utilization for a potential CPU bottleneck

FTP Performance Result Summary

FTP Performance Result Summary

Ù Achieveable data rates vary a lot

```
Depending on
             - VSE type of data
               (fixed or variable logical records)
             - type of DASDs
             - direction of transfer, etc
```

EDR ranges observed so far (1.3):

Effective Data Rate (KB/sec) ranges			
	FTP to VSE	FTP from VSE	Major impact/Comment
LIBR	340	470	DASD, network speed
POWER	(60) 115 - ...	(80) 290 - ...	(Improved, DBLK=7K) DBLK
VSAM ESDS (binary)	... ...	460 360 160	To S/390 To RS/6000 CLAW Via CLAW & T/R

Ù CPU resources required (1.3)

Vary also, depending on similar parameters

KI/KB values observed			
	FTP to VSE	FTP from VSE	Dependencies
LIBR	18.9 - 20.1e	11.9 - 13.3e	
POWER	(200) 85	(157) 45	(Improved)
VSAM ESDS	...	7.6 - 9.2	Conversion
- E.g. 20 KI/KB correspond to 2 MIPS per 100 KB/sec - Standard Dispatcher (SD) showed up to 10% lower values			

Impact of SD on data rate varies, depending on CPU-time share in total elapsed time (= CPU utilization)

Í Understanding all FTP figures and setting up hints is a challenge

FTP Performance Result Summary ...

I/A FTP Performance Results

Environments

- VSE/ESA 2.3.0/2.3.2 + TCP/IP level as indicated.
- Setup was basicly same as for the TN3270 runs (but w/o TPNS).
- TD was used by default, and a uni-processor. 'Old' SD was used for selected cases.
- All CPU-time (for TD) on VSE was obtained from QUERY TD. For SD, a H/W monitor was used.
- For TCP/IP, all default values were used. By intent the following ('ample') parameters were selected as working point:

```
SET WINDOW=32684 (default is 4096), also set in VM
DEFINE LINK CTC ... MTU=4096
```

Measurement runs and variations for 'I/A FTP'

- In TCP/IP for VM a PUT or MPUT was done (as FTP client and source) or GET or MGET (as FTP client and target).

We varied so far:

- the file(s) transferred: 1x10 MB or 10x1 MB TXT file (no conversion was included) For VSAM ESDS, the VSE.MESSAGES.ONLINE file (3M) was used
- the target location of the file in VSE: VD and Real Disk (9345 cached)
- the source location of the file in VM : VD and Real Disk

Sets of runs:

```
- FTP to and from VSE w/ LIBR (to/from a VSE library)

- FTP to and from VSE w/ POWER (to/from a POWER queue)

- FTP        from VSE w/ VSAM ESDS (from the VSE OME file)
```

FTP measurements with Batch partitions were also done

FTP Performance Results (LIBR)

FTP Performance Results (LIBR)

```
TCP/IP level UQ14494 (Service Pack E), runs were done 98-03-10/16.

     A  Total EDR (KB/sec, including overhead)
     |
     |                                             --- 1x10M
 600 -               592           593             ... 10x1M
     |               ---           ---
     -              |   |         |   |
     |              |   |         |   |
 500 -              |   |         |   |                   502
     |              |   |         |   |     448   (470)   ---
     -              |   |         |   |     ---    ---   |   |
     |              |   |         |   |    |   |  |   |  |   |
 400 -              |   |         |   |    |   |        |   |
     |              |   |         |   |    |   |  |   |  |...|
     -              |   |         |   |    |   |        |389|
     |        (320) |   |         |   |    |   |  |   |  |   |
 300 -         ---  |   |         |   |    |...|        |   |
     |        |   | |   |         |...|    |326|  |   |  |   |
     -   219        |...|         |288|    |   |        |   |
     |   ---  |   | |232|    220  |   |    |   |  |   |  |   |
 200 -  |...|       |   |    ---  |   |    |   |        |   |
     |  |207| |   | |   |   |...| |   |    |   |  |   |  |   |
     -  |   |       |   |   |216| |   |    |   |        |   |
     |  |   | |   | |   |   |   | |   |    |   |  |   |  |   |
 100 -  |   |       |   |   |   | |   |    |   |        |   |
     |  |   | |   | |   |   |   | |   |    |   |  |   |  |   |
     -  |   |       |   |   |   | |   |    |   |        |   |
     |  |   | |   | |   |   |   | |   |    |   |  |   |  |   |
     --------------------- ------------- ----------------------
      <--------- FTP to VSE -----------> <--- FTP from VSE --->
VSE Disk: R     E*    V       R     V   |   R       E*     V
VM Disk:  R     R     R       V     V   |   R       R      R

VSE CPU
 1x10M: >32%    -   >79%    >32%  >78%  |  >39%     -    >44%
 10x1M: >31%    -   >33%    >32%  >41%  |  >32%     -    >38%

VSE NPS  (improved meanwhile)
 1x10M: 0.310   -   0.273   0.313 0.289 |  0.336    -    0.350
 10x1M: 0.308   -   0.307   0.312 0.311 |  0.382    -    0.391

VSE KI/KB
 1x10M: 21.0    -   19.9    21.0  19.7  |  12.9     -    12.9
 10x1M: 21.4    -   20.7    21.1  20.7  |  14.3     -    14.3

Data Rates w/o overhead (from VM TCP/IP) were
                  up to 20% higher (1x10M)
                  up to 35% higher (10x1M)

* E is a (conservative) extrapolation from the 9345 (no DFW)
  to a faster and WRITE-cached I/O subsystem
```

FTP Performance Results (LIBR) ...

FTP Performance Results (LIBR) (cont'd)

FTP to VSE (LIBR)

```
 o  KI/KB varied between 19.9 and 21.4

 o  10x1M vs 1x10M (many small vs 1 big member):
    - needed up to 5% more CPU-time (member overhead in LIBR/TCP)
    - needed about twice the Elapsed time for VSE VD
             about 5% more Elapsed time for VSE real disk

 o  VSE Virtual Disk:
     - gave higher EDRs in all cases
     - gave much higher EDRs for 1x10M

 o  VM Virtual Disk:
     - did not change CPU-time in VSE (as expected)
     - gave only higher EDRs when VSE VD is used AND 10x1M


 Possible assessment of these data rates (based on available info)
 -----------------------------------------------------------------

 The biggest bottleneck was the slow 9345 for WRITEing

 o  When using VSE VD, the 1x10M case improved a lot,
    until it reached close to 100% CPU utilization.

    The 10x1M case improved much less, reason TBD
    (maybe VM CPU util, maybe VM directory access, etc).

 o  When using VM VD, only the 10x1M case improved,
    so this may be an indication for a slow VM directory access.

 o  It is unknown to what extent the CTC speed was exploited
    but it is at least as high as the 593 KB/sec case shows
```

FTP from VSE (LIBR)

```
 o  KI/KB varied between 12.9 and 14.3
 o  READ performance of the cached 9345 is much better than WRITE.
    Thus no VSE VD runs were done for FTP from VSE.
```

More Recent FTP Results (LIBR)

FTP Results (LIBR) for Service Pack F

```
Runs were done 98-04-08.

 - EDRs and CPU-times very similar to UQ14494 (Service Pack E)
 - NPS values reduced by about 35%:
            0.310 -> 0.190, or 0.336 -> 0.221
```

FTP Results (LIBR) for Service Pack I

```
TCP/IP level UQ22503, available 10/98, runs were done 98-10-09.

     A  Total EDR (KB/sec, including overhead)
     |
     |               613
 600 -               ---
     |              |   |                         --- 1x10M
     -              |   |
     |              |   |
 500 -              |   |                  483
     |              |   |                  ---
     -              |   |                 |   |
     |              |   |                 |   |
 400 -              |   |                 |   |
     |              |   |                 |   |  -> Up to 10% higher
     -        (340) |   |                 |   |           EDRs
     |         ---  |   |                 |   |
 300 -              |   |                 |   |  -> 5% to 10% less
     |   245  |   | |   |                 |   |           CPU-times
     -   ---        |   |                 |   |
     |  |   | |   | |   |                 |   |  -> Even lower NPS
 200 -  |   |       |   |                 |   |           values
     |  |   | |   | |   |                 |   |
     -  |   |       |   |                 |   |
     |  |   | |   | |   |                 |   |
 100 -  |   |       |   |                 |   |
     |  |   | |   | |   |                 |   |
     -  |   |       |   |                 |   |
     |  |   | |   | |   |                 |   |
     ----------------------------------- ----------------------
      <--------- FTP to VSE -----------> <--- FTP from VSE --->
VSE Disk: R     E*    V       R     V   |   R       E*     V
VM Disk:  R     R     R       V     V   |   R       R      R

VSE CPU: 35%    -    83%      -     -   |   41%     -      -
VSE NPS:0.126   -   0.169     -     -   |  0.108    -      -

 KI/KB: 20.0    -   18.9      -     -   |  11.9     -      -

- Data Rates w/o overhead were up to 10% higher.
```

FTP Performance Results 1.4 (LIBR)

FTP Performance Results 1.4 (LIBR)

```
TCP/IP level UQ48729 (1.4 ServPack A), runs done 2000-12-04.

     A  Total EDR (KB/sec, including overhead)
     |
     |               730                           --- 1x10M
 700 -               ---          |   |            ... 10x1M
     |              |   |         |   |
     -              |   |         |   |
     |              |   |         |   |
 600 -              |   |         |   |     620
     |              |   |         |   |     ---
     -              |   |         |   |    |   |         |   |
     |              |   |         |   |    |   |  |   |  |   |
 500 -              |   |         |   |    |   |         |   |
     |              |   |         |   |    |   |  |   |  |   |
     -              |   |         |   |    |   |         |   |
     |              |   |         |   |    |   |  |   |  |   |
 400 -              |   |         |   |    |   |         |   |
     |        (360) |   |         |   |    |   |  |   |  |   |
     -         ---  |...|         |   |    |...|         |   |
     |    265  |   | |310|         |   |    |345|  |   |  |   |
 300 -    ---       |   |         |   |    |   |         |   |
     |  |   | |   | |   |   |   | |   |    |   |  |   |  |   |
     -  |235|       |   |   |   | |   |    |   |         |   |
     |  |...| |   | |   |   |   | |   |    |   |  |   |  |   |
 200 -  |   |       |   |   |   | |   |    |   |         |   |
     /  /   / /   / /   /   /   / /   /    /   /  /   /  /   /
     -  |   |       |   |   |   | |   |    |   |         |   |
     |  |   | |   | |   |   |   | |   |    |   |  |   |  |   |
     --------------------- ------------- ----------------------
      <--------- FTP to VSE -----------> <--- FTP from VSE --->
VSE Disk: R     E*    V       R     V   |   R       E*     V
VM Disk:  R     R     R       V     V   |   R       R      R

VSE CPU
 1x10M: >40%    -   >76%                |  >39%     -
 10x1M: >25%    -   >34%                |  >32%     -

VSE NPS
 1x10M: 0.142   -   0.221               |  0.107    -
 10x1M: 0.166   -   0.216               |  0.143    -

VSE KI/KB
 1x10M: 14.0    -   14.6                |   9.1     -
 10x1M: 14.8    -   14.9                |   9.9     -

Data Rates w/o overhead were higher.

* E is a (conservative) extrapolation from the 9345 (no DFW)
  to a faster and WRITE-cached I/O subsystem
```

FTP Performance Results 1.4 (LIBR) ...

FTP Performance Results 1.4 (LIBR) (cont'd)

Conclusions for TCP/IP for VSE/ESA 1.4 ServPack A

```
Comparing TCP/IP for VSE/ESA 1.4 ServPack A vs 1.3 ServPack I ...
```

„ **Effective Data Rates (EDRs) increased by 10% to 30%**

```
      For the big member:

          - +10% (FTP to VSE, REAL Disk)
          - +20% (FTP to VSE, VIRT Disk)
          - +30% (FTP from VSE, REAL Disk)

      Similar improvements apply to smaller members
```

„ **CPU-time consumption decreased by about 25%**

```
      For the big member:

          - -30% (FTP to VSE, REAL Disk)
          - -23% (FTP to VSE, VIRT Disk)
          - -25% (FTP from VSE, REAL Disk)

      Similar improvements apply to smaller members
```

Results for FTP with Gateway TCP/IP

```
FTP to VSE via a network-owning (=gateway) TCP/IP partition
(ServPack A, 1x10M member, VSE/ESA Real Disk).

   o  Very similar EDR as in single TCP/IP case:
      255 KB/sec vs 265 KB/sec

   o  Higher total CPU-time consumption:
      23.4 KI/KB vs 14.0 KI/KB  (+65%!)

   o  CPU-time ratio between the FTP owning and the gateway
      TCP/IP:    1:0.39
```

Í **Try to avoid mass traffic thru a separate gateway TCP/IP**

FTP Performance Results (POWER)

FTP Performance Results (POWER)

```
TCP/IP levels: UQ16971 (1.3 ServPack F), runs done 1998-04-08.
               UQ48729 (1.4 ServPack A), runs done 2000-12-07.

     A  Total EDR (KB/sec, including overhead)
     |
     |                              288        |   |
     -                              ---        |   |
     |                             |   |       |   |
 250 -                             |   |       |   |
     |                             |   |       |   |
     -          --- 1x10M          |   |       |   |
     |          ... 10x1M          |   |       |   |
 200 -                             |   |       |   |
     |                             |   |       |   |
     -                             |   |       |   |
     |                |   |        |   |       |   |
 150 -                |   |        |   |       |   |
     |     129        |   |        |   |       |   |
     -    -.-.-       |   |        |   |       |   |
     |    |127|       |   |        |   |       |   |
 100 -    |   |       |   |        |   |       |   |
     |    |   |       |   |        |   |       |   |
     /    / F /       /   /        / F /       /   /
          | & |       |   |        | & |       |   |
     |    | A |       |   |        | A |       |   |
     ---------------------------- ------------------------
        <----- FTP to VSE ------> <---- FTP from VSE --->

VSE DBLK:  7K          23K       |   7K         23K

VSE CPU tot(POWER)
 1x10M:  >73%(49%) (F)           | >92%(6%) (F)
 1x10M:  >70%      (A)           | >76%     (A)
 10x1M:  >72%(48%) (F)           |

VSE NPS
 1x10M:  0.812 (F)               |  0.118   (F)
 1x10M:  0.857 (A)               |  0.102   (A)
 10x1M:  0.818 (F)               |

VSE KI/KB
 1x10M:   87.7 (F)               |   45.0 (F)
 1x10M:   82.8 (A)               |   40.5 (A)
 10x1M:   87.1                   |

- Data Rates w/o overhead (from VM TCP/IP) were up to 10% higher.
- FTP to VSE here fully dominated by slow uncached WRITEs of 9345s
- Bigger DBLKs would increase EDRs and reduce CPU consumption
```

FTP Performance Results (VSAM ESDS)

FTP Performance Results (VSAM ESDS)

```
TCP/IP level UQ16971 (Service Pack F), runs were done 98-04-08.


So far only FTP from VSE was done.

Besides BINARY, also EBCDIC to ASCII was selected.


      A  Total EDR (KB/sec, including overhead)
      |
      -                                     3222K ESDS (OME)
      |
  800 -
      |
      -
      |
  600 -
      |                             >460
      -                              ---    >390
      |                             |   |    ---
  400 -                             |   |   |   |
      |                             |   |   |   |
      -                             |   |   |   |
      |                             |   |   |   |
  200 -                             |   |   |   |
      |                             |   |   |   |
      -                             |   |   |   |
      |    |   |   |   |   |   |    |   |   |   |
      ---------------------------- ------------------------
         <----- FTP to VSE ------> <---- FTP from VSE --->

MODE:                            |  BIN   EBC/ASC

VSE CPU tot
 1x 3M:                          |  >21%     >23%

VSE NPS
 1x 3M:                          |  0.364    0.311

VSE KI/KB
 1x 3M:                          |    7.6      9.2

- Data Rate w/o overhead (from VM TCP/IP) for this short activity
  was 946 and 573 KB/sec
```

FTP Performance Hints -DASD Access-

FTP Performance Hints -DASD Access-

DASD access time is a major performance factor for FTP

Ù **General**

The 'better' a file locally is defined, and ...
the faster an I/O subsystem is ...

Í **the higher is the potential FTP data rate**
for single and multiple FTPs of any kind.

Ù **POWER related**

Check the priority of your POWER partition.

Single FTP:

Per POWER I/O to its Data File, only 1 DBLK is read/written

„ **Select a bigger DBLK size**

Use a DBLK size of about tracksize/2. This will speed up DASD access to the Data File, also for multiple FTPs

Multiple (concurrent) FTPs:

Or for single FTPs with high concurrent POWER spool activity.

POWER can concurrently issue 1 I/O to each data file extent, if on different logical DASDs (volumes)

„ **Select a bigger number of POWER Data File extents on different volumes**

This will increase the I/O concurrency, though it cannot be influenced on which extent certain data reside

Processor Resources Needed for FTP

Total CPU-time for FTP

The total CPU-time (CPUT) for FTP is (all in the TCP/IP partition) for a certain amount of transferred data (of certain type) is:

```
  CPUT in FTP application   =f(file type, blocking ...)
+ CPUT in TCP/IP layers     =f(TCP/IP parms, network ...)
-------------------------
= total CPU-time for FTP
```

The CPU-time per KB transferred likewise varies with these parameters.

To be basicly processor speed independent, it may be appropriate to roughly use

```
               CPU-time =  pathlength / MIPS.
```

'KI per KB' observed values roughly were (1.4)

```
            15 / 10 K instructions (KI) per KB   (to/from LIBR)
            80 / 40 K instructions (KI) per KB   (to/from POWER)
```

Processor Capacity Needed for FTP

This 'KB per KI' value determines, together with the achieved actual total EDR, the actually used/required CPU utilization when FTP is actually running:

```
                   'KI per KB' x  EDR
   CPU utiliz.  = -------------------
                    KIPS of 1 engine
```

EDR is the rate for a single or multiple FTP transfers.
Use this formula multiple times if KI per KB differs.

Assumed example (1.4):

```
      15 KI/KB (LIBR)
      300 KB/sec total FTP rate,  on a (approx.) 20 MIPS processor:
```

```
                15 KI/KB  x 300 KB/sec
CPU utiliz. = ------------------------ = 0.225 = 23%
                    20000 KIPS
```

FTP VS-Capacity

FTP VS-Capacity

The following data represent measurements of the virtual storage requirements for FTP daemons within a TCP/IP partition.

TCP/IP 1.4 ServPack A was used and FTP daemons defined on top of our standard TCP/IP setup.

Cost of FTP Daemons

	10 daemons -24	 -31	Per daemon -24	 -31
TCP/IP GETVIS	3104K	40K	310K *)	4K

- No VS used for FTP daemons in SVA

Conclusion:

Rough estimate for FTP VS-Capacity

Considering FTP daemons within TCP/IP partition.

Based on *) above

```
Max. #FTP daemons = (remaining GETVIS-24) / 310K
```

'Remaining GETVIS-24' is that GETVIS which is available in your TCP/IP partition, IF you would start it again, BUT w/o any FTP daemons defined.

Here, the remaining GETVIS-24 was about 10M, resulting in (theoretically) 32 FTP daemons for 'interactive' FTP.

Batch FTP Performance Results

Batch FTP Performance Results

Environments

- same as for the I/A FTP runs

Measurements and variations for 'FTP with Batch'

- FTP was initiated in a VSE Batch partition, via // EXEC FTP or FTPBATCH
- The 10M file was used in BINARY mode, except 1 run with 1M (to determine batch part. overhead)
- The target location was a VSAM ESDS file
- All FTPs were from VM (real disk) to VSE via real CTC
- So far, only Service Pack K was used

We varied so far:

- the type of partition (static/dynamic) for
  - TCP/IP for VSE/ESA
  - Batch FTP (// EXEC FTP)
  - FTPBATCH (// EXEC FTPBATCH)
- the location of the target file in VSE
  - VSE Virtual Disk
  - VSE Real Disk (9345 only read-cached)
- the VSAM CI-size was varied once (8K instead of 4K CIs)
- Batch FTP and FTPBATCH were run once also on a 2-way
- Multiple concurrent FTPs (not yet)

Sets of runs:

- FTP to VSE with different partition types

Runs were done 99-07-19 and 99-07-21.

Batch FTP Performance Results ...

FTP with Batch, Performance Observations

CPU-time Overhead vs I/A FTP

'Overhead' here includes

- Batch partition initiation (incl. Job Control etc).

and

- Movement of data between FTPBATCH and TCP/IP partition.

		Static part.	Dynamic Part.
Batch FTP	CPU-time ET (rough)	1.45 sec 6 sec	1.62 sec 7.5 sec
FTPBATCH	CPU-time ET (rough)	2.38 sec 11 sec	2.45 sec 12.5 sec

- All values here are deltas to I/A FTP
- Emphasis here is on principal deltas, not typical values
- Here, the total ET varied around about 30 sec.

Dynamic Partition overhead is slightly higher than static

As expected (of interest only for small file transfers).

FTPBATCH has data movement overhead vs Batch FTP

Refer to next foils

Batch FTP Performance Results ...

FTP with Batch, Performance Observations (cont'd)

Data Rates when Transfer was started

Here mostly the base measured cases are shown.

	'Overall EDR' Transfer KB/sec Real 9345	 Virt.Disk	File I/O KB/sec Real 9345	 Virt.Disk
I/A FTP	639	930e	682	1462e
Batch FTP	639	930	682	1462
FTPBATCH				
stat-stat	511		682	
dyn -dyn	538*1	787	682	1462
dyn -dyn	568*2	-	731*2	-

- Total Transfer time includes File-I/O time.
- File I/O rates to Virtual Disk are much higher than to non-WRITE-cached 9345.

*1 Higher rate if both FTPBATCH and TCP/IP run in a dynamic partition.

*2 Higher rate to 9345s with CISIZE=8K (vs 4K)

- EDRs for I/O Subsystems with DFW are expected to be about 30% higher than shown above for 9345s, since much faster for seq. WRITEs.

Same rates as for I/A FTP, except Transfer rate seen by FTPBATCH

Data rate is measured in FTPBATCH partition and includes move to TCP/IP partition, plus TCP and IP layers there.

Overall EDRs for FTP with batch depend on share of partition overhead

Overall EDRs for (single) FTPBATCH are about 15% lower here than for Batch FTP

Cont'd

Batch FTP Performance Results ...

FTP with Batch, Perf. Observations (cont'd)

„ **CPU-time and overall EDRs (Real Disk)**

FTP of a 10M file (BINARY) to VSE ESDS on Real Disk (1-way)

TCP-Batch	CPU-time	KI/KB	EDR (via ET)
I/A FTP	10.167 sec	13.9	> 492 KB/sec
Batch FTP stat-stat	11.619 sec	15.9	> 380 KB/sec
stat-dyn	11.791 sec	16.1	> 362 KB/sec
dyn -dyn	11.359 sec	15.5	> 391 KB/sec
*1 dyn -dyn	-	15.0e	500eKB/sec *1
FTPBATCH stat-stat	12.550 sec	17.1	> 300 KB/sec
stat-dyn	12.622 sec	17.2	> 305 KB/sec
dyn -dyn	12.375 sec	16.9	> 303 KB/sec
*1 dyn -dyn	-	15.5	471 KB/sec *1

```
 -   CPU-time and KI/KB include variable partit. overhead
 -   All EDRs here apply to the slow 9345 disks
 *1  Without JCL overhead, from 10M run -1M run.
 e   estimate
 -   The VSE JA CPU-time in the Batch partition
     (which is a part of the total CPU times shown above)
     were (approx):   5% for Batch FTP
                     70% for FTPBATCH
```

í **No specific impact seen for Batch FTP depending on partition type**

í **FTPBATCH with slightly higher CPU-time and with lower EDR**

Batch FTP Performance Results ...

FTP with Batch, Perf. Observations (cont'd)

„ **CPU-time and EDRs (Virt. Disk, Single Stream)**

FTP of a 10M file (BINARY) to VSE ESDS on Virtual Disk.
All values include (dynamic) partition overhead.

Single Stream	CPU-time	KI/KB	NPS	EDR (via ET)
Batch FTP 1-way	11.025 sec	15.05	.265	> 495 KB/sec
2-way	11.460 sec	15.67	.267	> 468 KB/sec
FTPBATCH 1-way	12.832 sec	15.54	.318	> 375 KB/sec
2-way	15.027 sec	20.54	.353	> 386 KB/sec

- NPS is lower for real disk and w/o partition overhead

í **A 2nd engine**
- does not help at all for Batch FTP
5% lower data rate, 4% higher CPU-time
- hardly helps for single stream FTPBATCH
3% higher data rate, at cost of 17% higher CPU-time

„ **CPU-time and EDRs (Virt. Disk, Mult. Stream)**

Figures to be provided, no runs done so far

í **FTPBATCH file transfers**

can be better workload balanced (controlled) via PRTY/PRTYIO

Especially needed when only 1 TCP/IP partition is used

can run concurrently and thus achieve a higher sum of FTP EDRs

Especially beneficial when FTPs are to/from multiple real disks
(where a single File I/O routine would be a bottleneck)

allow to exploit >1 processor engines

TCP4VSE Performance Results - GPS

PART H.

TCP4VSE Performance Results - GPS

Ù **GPS VS-Capacity**

GPS VS-Capacity

GPS VS-Capacity

The following data represent measurements of the virtual storage requirements for GPS daemons within a TCP/IP partition.

TCP/IP 1.4 ServPack A was used and GPS daemons defined on top of our standard TCP/IP partition setup.

í **Cost of GPS Daemons**

GPS TCP/IP GETVIS Requirements

	10 daemons -24	 -31	Per daemon -24	 -31
(QUEUING=DISK)	1172K	212K	117K *)	21K
(QUEUING=MEMORY)	128K	212K	13K	21K

- No VS used for GPS daemons in SVA
- The bigger values for QUEUING=DISK is caused by LIBR control blocks and nonshared LIBR buffers
- Note that 10K are not freed at DELETE of a GPS daemon, but are reused instead for other GPS daemons
- For QUEUING=MEMORY, traffic dependent GETVIS has to be added

Conclusion:

í **Rough estimate for GPS VS-Capacity**

Based on *) above

```
Max. #GPS daemons = (remaining GETVIS-24) / 117K
```

'Remaining GETVIS-24' is that GETVIS which is available in your TCP/IP partition, IF you would start it again, BUT w/o any GPS daemons defined.

Here, the remaining GETVIS-24 was about 10M,
resulting in (theoretically) 85 GPS daemons (QUEUING=DISK).

í **Try to use QUEUING=MEMORY for 'small printouts only' printers**

SSL Performance View

PART I.

SSL Performance View

Overview

SSL Performance View

- References/Glossary
- Principal Functions of SSL
- SSL/TLS General Implementation
- Cryptography Summary
- TLS Layer and Secure Sessions
- SSL in VSE/ESA
- Performance of SSL in VSE/ESA

References

References

- 'SSL and TLS Essentials: Securing the Web'
  by Stephen A. Thomas, John Wiley, March 2000, 197 pages
- 'Internet Security'
  by Don Stoever, Connectivity Systems Incorporated (CSI)
  via www.tcpip4vse.com, under 'Available Presentations'
- 'TCP/IP Tutorial and Technical Overview'
  by Martin Murhammer et al, IBM Redbook
  via www.redbooks.ibm.com
- 'Secure Socket Layer and Transport Layer Security'
  by Liisa Erkomaa, Helsinki University,
  via www.tml.hut.fi/studies/tik-110.350/1998/essays/ssl.html
- 'RSA Laboratories' FAQs about Today's Cryptography',
  Version 4.1, 05/2000
  via www.rsa.com
- 'The TLS Protocol V 1.0', Request for Comments: 2246
  Network Working Group, 01/1999.
  via www.ietf.org
- 'Secure Hash Standard' FIPS PUB 180-1, 05/1993.
  via www.nist.gov
- 'Introduction to SSL'
  via docs/iplanet.com/docs/manuals/security/sslin/index.html
- home.netscape.com/security/index.html
- 'IPSec, The New Security Standard for the Internet,
  Intranets, and Virtual Private Networks'
  by N. Doraswamy and D. Harkins (ISBN 0-13-011898)
- 'SSL for VSE',
  Documentation by Connectivity Systems Inc. (CSI)
  by Don Stoever, 12/2000.
  Also presented at the Tech Conf 05/2001,
  Jacksonville, Florida

Glossary

Glossary

Terms

Authentication	The positive identification of a network entity e.g. server, client or user. Within SSL the server and client Certificate verification process
Cipher	An algorithm or system for data encryption or a cryptographic method
Digest	The result (a special mathematical summary) of a special cryptographic function called hashing. Also called a 'fingerprint'
Digital Certificate	A data record used for authentication. Contains X.509 information pieces about its owner and the signing Certificate Authority, plus the owner's public key and the signature made by the Certification Authority
Certificate Authority	A trusted third party providing authentication for network entities: Allocates, certifies and guarantees that a certain public key belongs to a certain owner.
Identification	The process of verification of the correct Digital Certificate of the other side
Message Digest	A hash of a message which can be used to verify that contents of a message has not been altered
Plaintext	The unencrypted text of a message
Ciphertext	The encrypted text of a message

Glossary ...

Glossary (cont'd)

Abbreviations

SSL	Secure Sockets Layer Netscapes secure sockets layer protocol, often also used for it's successor TLS
TLS	Transport Layer Security Successor of the SSL security protocol, set up by the IETF in an RFC
IETF	Internet Engineering Taskforce Open international community of networking people. Pushes setup and acceptance of Internet standards
PKI	Public Key Infrastructure All components, processes and concepts used in connection with public keys
X.509	A standard which essentially defines a format for digital certificates
RC4	Rivest encryption Ciphers A stream encryption cipher developed by Rivest
CBC	Cipher Block Chaining A cipher mode where every plaintext block encrypted is first exclusive-ORed with the previous ciphertext block.
RSA	Rivest Shamir Adleman A very widely used public-key algorithm that can be used for either encryption or digital signing. It is based on the factoring problem with prime factors
DES (3DES)	Data Encryption Standard (Triple DES) An encryption algorithm (... applied 3 times). DES is a block cipher with a 56 bit key and an 8 byte block size. Developed by IBM and the US Government

Glossary ...

Glossary (cont'd)

DSA	Digital Signature Algorithm Part of the digital authentication standard
MD5	Message Digest 5 A secure hashing algorithm developed by Ron Rivest. Converts an arbitrarily long data stream into a digest of fixed size (16 byte)
SHA	Secure Hash Algorithm Proposed by the US National Institute of Science and Technology (NIST): creates a 20-byte digest. SHA-1 is a technical revision of SHA
MAC	Message Authentication Code A 1-way hash computed from a message and some secret data. Targetted to detect whether a message has been altered
HMAC	A certain type of keyed MAC (keyed hash inside keyed hash), cryptographically stronger. Can be used with any iterative cryptographic hash function, e. g. together with MD5 or SHA-1, as in TLS. Refer to RFC 2104

Trademarks

Many names used in this part on SSL are trademarks or registered trademarks of their respective owners, e.g. RSA Security Inc. and others

Principal Functions of SSL (TLS)

Principal Functions of SSL (TLS)

Originally developed by Netscape, universally accepted (de facto standard) and now further developed under the name TLS by the IETF.

Ù **Confidentiality**

Keeping secrets, protect against eavesdropping (snooping)

Í **Cipher messages**

E.g. RC5, DES or 3DES

Ù **Integrity**

Verifying information, protecting against alteration

Í **Use a hash function (similar to checksum)**

E.g. MD5, SHA

Ù **Authentication**

Proving identity (server and client/user), protect against forgery (falsification) and masquerade (hiding)

Í **Add a secret 'item' in the message**

E.g. MAC or HMAC type use of a hash algorithm

Ù **Nonrepudiation**

Making sure that sender cannot falsely deny that he sent the message

Í **Digital signatures**

E.g. DSA

SSL(TLS) General Implementation

SSL(TLS) General Implementation

In the context of this document, SSL is also used as a general term comprising also the TLS level of security.

Ù **SSL implements these functions using Cryptography**

Refer to separate suite of charts

Ù **A new separate protocol layer is used**

Refer to the following chart

Important:

Please differentiate between SSL Encryption and Firewall Security

SSL Protocol Layer

TCP/IP with SSL Protocol Layer

```
     Layer
 ----------------                 ------------------------------
|                |               |                              |
|                |               | TELNET            SNMP       |
|                |               | FTP(TCP)          FTP(UDP)   |
|  Application   |    MESSAGE -> | HTTP(WWW)         NFS(RPC)   |
|                |               | SMTP(E-mail)      ...        |
|                |               | ...             |            |
 ---------------                  ------------------            |
|/              /|               |/               /|            |
|/              /|               |/ SSL/TLS layer /|            |
|/ SSL/TLS layer/|               |/ ============= /|    *1      |
|/              /|               |/ (Encryption)  /|            |
|/              /|               |/               /|            |
 ---------------                  ------------------------------
|                | UDP DATAGRAM->|                 |            |
|                | TCP SEGMENT-> |                 |            |
|  Transport     |               |    TCP          |  UDP       |
|                |               |(connection      |(connection-|
|                |               |       oriented) |      less) |
 ----------------                 ------------------------------
|                |               |                              |
|  Internetwork  | IP DATAGRAM-> |             IP               |
|                |               |     (connectionless)         |
|                |               |                              |
 ---------------                  ------------------------------
|  Link          |               |                              |
| (Device Driver)|               | Token-Ring, Ethernet, FDDI,  |
|                |    FRAME ->   | ATM                          |
 ................                 ..............................
|                |               |                              |
|  Hardware      |               |                              |
 ----------------                 ------------------------------
```

*1 UDP not supported by SSL/TLS

- SSL/TLS layer appears
    - to TCP as a TCP application
    - to the SSL application as previously the TCP layer

Cryptography (Summary)

Cryptography (Summary)

Ù **Secret Key Cryptography**

Both parties (only they) know the same secret key

'Symmetrical encryption'

```
                    Secret key
                  /             \
                /                 \
 ---------     |    ----------     |     ---------
|Plaintext|    V   |#%&*(*H&¢ |    V    |Plaintext|
|         |------->|          |-------->|         |
|         |        |          |         |         |
|         |        |          |         |         |
 ---------          ----------           ---------
```

Stream ciphers (e.g. RC4)

Continuous enciphering

Block ciphers (e.g. RC5, DES, 3DES)

Operates on blocks, often 64 bit.
More often used than stream ciphers.
RC5 e.g. has a key length up to 2040 bit

Other examples of symmetrical encryption:
- Blowfish
- encrypt/decrypt data within a single PC

Size of key is important

í **Problem: to safely exchange the secret keys (henn and egg problem)**

Cryptography (Summary) ...

Cryptography (Summary) (cont'd)

Ù **Public Key Cryptography**

Separate keys for encryption and decryption

'Asymmetrical encryption'

```
          Public key          Private key

              |                    |
 ---------    |    ----------      |     ---------
|Plaintext|   V    |#%&*(*H&¢ |    V    |Plaintext|
|         |------->|          |-------->|         |
|         |        |          |         |         |
|         |        |          |         |         |
 ---------          ----------           ---------
```

Only 1 key needs to be secret (private key), the other 'can/should be published'

Usually the public key is for encryption, the private key for decryption.

- ENcryption and DEcryption never can be done with only 1 key!
- PRIVATE and PUBLIC key belong together.
  They are mathematically dependent (a secret relation),
  BUT the exact relation is unknown to all
  (except maybe the owner of the private key)
- PGP (Pretty Good Privacy) e.g. uses asymmetrical encryption

í **Problem: this encryption is very complex**

Symmetric ciphers are many times faster

Cryptography (Summary) ...

Cryptography (Summary) (cont'd)

Ù **The RSA algorithm (Rivest Shamir Adleman)**
(based on prime factors)
also works in reverse order

Information encrypted with private key can be decrypted with the corresponding public key

'Reverse Public Key algorithm'

Patent expired 09/2000

Ù **Combining Secret and Public Key Cryptography**

'Use public key cryptography to safely exchange secret keys to be used in (the rest of) the session'

Refer to the example on the next foil

Comparing Ciphers

Ù **The strength of a cryptosystem is given by**

its effort to encrypt/decrypt
the required effort to crack a key

Ù **(Effective) Key length is of prime importance**

- 40 bit keys (no more safe enough)
- 56 bit
- 64 bit
- 112 bit

Public Key Principle with RSA

Public Key Principle with RSA

This is a very high level example for a handshake (using asymetrical encyption) to initiate a secure communication which then uses symmetrical encryption.

```
                                            BILL (server)
                                     -------------------------
      FRED (client)                 | Creates keys.            |
  ------------------------          |                          |
 | Wants a secure commu-  |         | Keeps private key secret|
 | nication with BILL.    |         | Publishes public key     |
 |"What's your public key?| Hello    -------------------------
 | My protocols are ..."  |------->  -------------------------
  ------------------------          |                          |
                            Public  | Sends his own public key|
  ------------------------ <------- | + protocol to be used    |
 | Knows BILLs public key |  key     -------------------------
  ------------------------          | Encrypts a known info    |
  ------------------------          | with own private key     |
 |Decrypts msg with BILLs |<-=-=-=-  -------------------------
 |public key.             |Signature
 |If identical to known   |
 |info, it WAS encrypted  |
 |with BILLs private key: |
 |    It's BILL!          |
  ------------------------
  ------------------------           -------------------------
 | Encrypts 1st msg with  | 1st msg | Decrypts 1st msg with    |
 | BILLs public key       |=======> | his own private key      |
  ------------------------           -------------------------
  ------------------------           -------------------------
 | Msg exchange           |         | Msg exchange             |
 | with common SECRET key |<.......>| with common SECRET key   |
  ------------------------           -------------------------
```

- BILLs PUBLIC key is only used by FRED,
  (to encrypt handshake msg and to decrypt signature for
  authentication)
  BILLs PRIVATE key only used by BILL.

- You simply must believe the properties of RSA!

- The 1st encrypted message usually contains the SECRET key
  (generated by FRED via random numbers)
  for a symmetrically encrypted session

- Public keys may also be obtained from a public key server

- Client authentication by server is optional (not shown here).
  Then also FRED would need to own a public and private key

TLS Layer

TLS Layer

Ù **Characteristics of TLS**

More separation and formal interface between the handshaking process and the record layer:

```
 --------------------------------
|                                |
| TLS Handshake Protocol         |
|                                |
| - Asymmetric or Private Key    |
|                                |
|................................|
|                                |
| TLS Record Protocol            |
|                                |
| - Symmetric Encrypt/Decryption |
|   (keys unique for connection, |
|    based on a secret negotiated|
|    by the handshake protocol)  |
|                                |
| - Secured Hashing Computations |
|                                |
 --------------------------------
```

Ù **TLS 1.0 is based on the SSL 3.0 protocol**

Differences are not dramatic, but significant.

TLS 1.0 clients are not yet widely available.

From IETF, SSL V3.0 description

```
"The (TLS) Record Protocol
 takes a message to be transmitted,
  - fragments the data into manageable blocks,
  - optionally compresses the data,
  - applies a MAC,
  - encrypts and transmits the results.

 Received data is
  - decrypted,
  - verified,
  - decompressed and
  - reassembled
  - then delivered to higher level clients"
```

Secure SSL/TLS Communication

A secure SSL/TLS communication/session ...

„ **uses a secured port**

„ **starts with 'handshake' (asymmetrical encr.)**
(with negotiation and agreement on cipher suite)

The PKI certificate is sent, also containing the public key of the server.
The client uses the public key to securely encrypt a secret random value.

The secret random value is used to create keys for encrypting and authenticating data that flows over the connection.
The key values used are generated unique to that session and (usually) are never reused.

„ **exchanges data with symmetrical encryption**

Í **combines both cryptographical methods**

Cipher Suite

Defines the various cryptographic options:

Key exchange method/algorithm

Also includes identification with X.509v3 certificates

RSA

Data encryption method/algorithm

1 out of NULL, RC4, DES, RC5, 3DES

Message authentication method/algorithm

MD5 or SHA message hashing, usually applied with HMAC

More on TLS

More on TLS

Ù **Hash Functions in TLS**

Hash Function	Hash Size (Digest)	Padding Size
MD5	16 byte	1 to 512 bit
SHA-1	20 byte	1 to 512 bit

Both hash functions are used in TLS with the HMAC mechanism for message authentication. Refer to RFC 2104.

SHA-1 appears to be cryptographically stronger than MD5, but needs more CPU cycles

Ù **Relative Efficiency of TLS**

From RFC 2246:

'Cryptographic operations tend to be highly CPU intensive, particularly public key operations'

This in mind, resulted in ...

Ù **TLS Session Caching**

- Optional session caching scheme is included to improve performance for session establishment:

  Reuse of cryptographic parameters of a previously established session between the same client and server.

  The session cache size may be controlled by the SSL application.

  If set to 0, no session parameters will be reused.

External vs Internal SSL Implementation

External vs Internal SSL Implementation

Ù **External SSL**

```
                                   VSE/ESA Host
                              --------------------
        'SSL server'         |        |           |
        -----------          |                    |
       | e.g.|     |         | TCP/IP |           |
       |           |         |                    |
  SSL  | MS  |     | non-SSL | w/o SSL|           |
<=====>| IIS       |<------->|                    |
       |     |     |         |        |           |
        -----------          |                    |
      e.g. Windows NT        |        |           |
                              --------------------
```

External SSL servers (which also can be used as a front-end to VSE/ESA are available e.g. from

```
- Data21      (IpBridge/Secure)
- IntelliWare (iC.Y.A (tm))
- BSI         (IpServer)
- Renex       (eZGATE)
```

Ù **SSL for VSE/ESA is an 'internal' SSL implementation**

How Ciphered Data can be Cracked

How Ciphered Data can be Cracked

This is a very simple and global view of this subject.

The following type of attacks are possible:

Ù **Plaintext (source text) is partly known**

Spy can use the knowledge by trying to decipher known parts, and then try use the info on the key to decipher the unknown parts.

Ù **Plaintext (source text) is not known**

Spy must search for specific patters, which recur, e.g. headers in an e-mail. Based on this the key is tried to be found.

Ù **The spy had access to the source text**

In this case the spy had the possibility to cipher the source text with an unknown key. With that information can be collected to identify the key.

Ù **The man in the middle**

During the exchange of the keys for asymmetrical encryption, a third party intercepts all messages and communicates by separate own keys both with the sender and the receiver.

SSL for VSE/ESA Implementation

SSL for VSE/ESA Implementation

Also referred to as 'SSL/VSE' or 'SSL for VSE'.

Implemented as a key protected feature to TCP/IP for VSE/ESA 1.4 ServPack B.

Note that the IBM TCP/IP 1.4 ServPack 'B' equivalent PTF does not yet support SSL, in contrast to the CSI version.

Ù **SSL 3.0 and TLS 1.0 (RFC 2246)**

Implementation is done via Assembler, NOT using the RSA package written in C.

1. **Key Exchange Algorithm (plus Identification)**

   RSA only (512 or 1024 bit)

   Plus X509v3 digital certificates used

2. **Encryption Algorithm**

   DES (40/56 bit keys)
   3DES (168 bit, effective as 112 bit)

3. **Message Hash Algorithm**

   Also used during pure data exchange

   MD5 (RFC 1321)
   SHA-1 (recommended)

   Either one used with HMAC (RFC 2104) for message authentication

SSL for VSE/ESA Implementation ...

Ù **New Secure Socket Layer API**

The SSL APIs are compatible to the OS/390 SSL API.

```
E.g.       gsk_initialize( )
           gsk_get_cipher_info( )
           gsk_uninitialize( )
           gsk_secure_soc_init( )
                -"-      _read( )
                -"-      _write( )
                -"-      _reset( )
                -"-      _close( )
```

Refer to the SSL for VSE description by CSI, and to 'OS/390 SSL Programming Guide and Reference' SC24-5877.

Ù **CryptoVSE API**

In addition, also a CryptoVSE API is provided.

```
E.g.       cry_initialize( )
           cry_des_encrypt()
           cry_des_decrypt()
           cry_sha_hash()
```

Can be used in case cryptographic functions are directly to be done by the application.

Ù **SSL-enabled TCP applications (status 04/2001)**

Note that
- always a corresponding SSL enabled client is required
- client authentication in VSE is to be provided if needed

In a fully transparent way:

- Telnet (TN3270)
- HTTP Web Server
- Existing TCP Socket Applications

Via direct use of the new SSL sockets:

- Any new TCP SSL socket application
- EZASMI Assembler macro I/F
- SSL enabled VSE Java Beans Connectors (VSE/ESA 2.6). A single VSE Connector Server can only be started with or without SSL
- POWER 'PNET SSL' (VSE/ESA 2.6).
- CWS with SSL (VSE/ESA 2.6)

SSL/VSE Commands

SSL/VSE Commands

Ù DEFINE SSL Daemon

1 SSL daemon for every transparently secured TCP port, in order to intercept the messages via that communication port

```
DEFINE TLSD,ID=...,PORT=xxx,PASSPORT=yyy,CIPHER=nn,
            MINVERS=0300|0301
```

```
CIPHER=nn    Specifies acceptable cipher suites
                01 = RSA512_NULL_MD5
                02 = RSA512_NULL_SHA
                08 = RSA512_DES40CBC_SHA
                09 = RSA1024_DESCBC_SHA   (for US?)
                0A = RSA1024_3DESCBC_SHA  (for US?)
                62 = RSA1024_EXPORT_DESCBC_SHA   (TLS 1.0)

MINVERS      Specifies the SSL version
                0300 = SSL 3.0
                0301 = TLS 1.0 (not yet av. for many clients)

PORT=xxx     Specifies the port that is to be secured.
             This port is being listened by this SSL daemon

PASSPORT=yyy Specifies the port that the SSL daemon has to
             pass the decyphered incoming messages to.
```

For applications using the new SSL API, PORT and PASSPORT are identical.

Ù QUERY TLSD

Displays info on the corresponding DEFINE TLSD command

SSL/VSE Performance Related Parameters

SSL/VSE Performance Related Parameters

The following table shows those settings in TCP/IP for VSE which are performance relevant (CPU-time) for SSL, together with the type of SSL activities a parameter has influence on.

	Type of SSL Activity	
SSL/VSE Parameter/setting	Handshaking (session overhead)	Data exchange (message overhead)
Key Exchange Algorithm		
RSA512	X	-
RSA1024	X	-
Encryption Algorithm		
NULL	-	X
DES40CBC (40)	-	X
EXPORT_DESCBC (40)	-	X
DESCBC (56)	-	X
3DESCBC (168)	-	X
Hash Algorithm		
MD5	X	X
SHA	X	X
Session Caching	X	-
Message Length	-	X1

X1 Data exchange ('message') overhead is proportional to bytes/msg (apart from padding)

Note that the CPU-time overhead caused by SSL (both the session and the message overhead) is in

- the TCP/IP partition (in case of the SSL Daemon)
- the application partition (in case of native SSL API usage)

SSL/VSE Performance Results

SSL/VSE Performance Results

Environment

- VSE/ESA 2.6 + TCP/IP 1.4 SPack 'B' with several levels of fixes and few remaining traces, 1 VSE processor (plus DEBUG OFF with a DEBUG=YES generated supervisor) under VM/ESA as V=V guest ('VSECON') on a 9672-RX4 10-way ('BOEVMSPA') with roughly 45 MIPS per engine.
- ECHO transaction (CONNECT, SENDRECEIVE small/big message) (Was used, though it was not available w/o encryption)
- Navigator txns (CONNECT SEND/RECEIVE small/large message)
- Various cipher suites (0A, 09, 02, 01 and '00')
- VSE CPU-times measured via QUERY TD,INTERNAL (so far no RTs or resources outside VSE measured in detail)
- Every measurement repeated several times

Measurement Results

TBD

Measurement Conclusions

TBD

Summary

Summary

Ù TCP/IP performance tuning is not easy

„ **Many S/W parameters included and potentially big networks**

„ **Careful setup and analysis may be required**

Ù We need to further improve TN3270 performance

„ **CPU-time overhead per txn**

„ **Partition capacity (via 31-bit exploitation)**

Resolved by Service Packs J and K

Ù FTP processor requirements and FTP data rates - have been already improved - will be enhanced further

Multiple FTP data rates improved via FTPBATCH

Ù We always try to improve/extend the tuning guidelines and documentation

Naturally, first priority here is the performance of the product

```
EOD          End Of Document
HAND         Have A Nice Day
```