VMS Software

VSI OpenVMS

VSI TCP/IP Services for OpenVMS Tuning and Troubleshooting

Document Number: AA-RN1VB-TE

Publication Date: January 2022

Revision Update Information: This is a new manual.

Operating System and Version: VSI OpenVMS Integrity Version 8.4-2
VSI OpenVMS Alpha Version 8.4-2L1

VMS Software, Inc. (VSI)
Burlington, Massachusetts, USA

VSI TCP/IP Services for OpenVMS Tuning and Troubleshooting

Legal Notice

Preface

The VSI TCP/IP Services for OpenVMS product is the VSI implementation of the TCP/IP networking protocol suite and internet services for VSI OpenVMS Alpha and VSI OpenVMS VAX systems.

TCP/IP Services provides a comprehensive suite of functions and applications that support industry-standard protocols for heterogeneous network communications and resource sharing.

This manual provides system and network managers with information they need to identify and resolve problems. This manual is best used in conjunction with the *VSI TCP/IP Services for OpenVMS Management* manual.

See the *VSI TCP/IP Services for OpenVMS Installation and Configuration* manual for information about installing, configuring, and starting this product.

1. About VSI

VMS Software, Inc., (VSI) is an independent software company licensed by Hewlett Packard Enterprise to develop and support the OpenVMS operating system.

VSI seeks to continue the legendary development prowess and customer-first priorities that are so closely associated with the OpenVMS operating system and its original author, Digital Equipment Corporation.

2. Intended Audience

This manual is for OpenVMS or UNIX system managers who are experienced in troubleshooting complex software products. This manual assumes a working knowledge of TCP/IP networking, TCP/IP terminology, and familiarity with TCP/IP Services. Always read all the current product documentation before attempting to resolve any problems.

3. Document Structure

This manual contains the following two chapters and appendix:

- Chapter 1 describes how to determine the cause of networking problems. It introduces some tools useful in monitoring and diagnosing these problems.
- Chapter 2 describes the TCP/IP subsystem attributes that you can adjust to improve network performance.
- Appendix A describes the tools available for isolating and resolving network problems.

4. Related Documents

Table 1 lists the documents available with this version of TCP/IP Services.

Table 1. VSI TCP/IP Services for OpenVMS Documentation

Manual	Contents
VSI TCP/IP Services for OpenVMS Concepts and Planning	This manual provides conceptual information about TCP/IP networking on OpenVMS systems, including general planning issues to consider

Manual	Contents
	before configuring your system to use the TCP/IP Services software. This manual also describes the manuals in the TCP/IP Services documentation set and provides a glossary of terms and acronyms for the TCP/IP Services software product.
VSI TCP/IP Services for OpenVMS Release Notes	The release notes provide version-specific information that supersedes the information in the documentation set. The features, restrictions, and corrections in this version of the software are described in the release notes. Always read the release notes before installing the software.
VSI TCP/IP Services for OpenVMS Installation and Configuration	This manual explains how to install and configure the TCP/IP Services product.
VSI TCP/IP Services for OpenVMS User's Guide	This manual describes how to use the applications available with TCP/IP Services such as remote file operations, email, TELNET,TN3270, and network printing.
VSI TCP/IP Services for OpenVMS Management	This manual describes how to configure and manage the TCP/IP Services product.
VSI TCP/IP Services for OpenVMS Management Command Reference	This manual describes the TCP/IP Services management commands.
VSI TCP/IP Services for OpenVMS Management Command Quick Reference Card	This reference card lists the TCP/IP management commands by component and describes the purpose of each command.
VSI TCP/IP Services for OpenVMS UNIX Command Equivalents Reference Card	This reference card contains information about commonly performed network management tasks and their corresponding TCP/IP management and Tru64 UNIX command formats.
VSI TCP/IP Services for OpenVMS ONC RPC Programming	This manual presents an overview of high-level programming using open network computing remote procedure calls (ONCRPCs). This manual also describes the RPC programming interface and how to use the RPCGEN protocol compiler to create applications.
VSI TCP/IP Services for OpenVMS Sockets API and System Services Programming	This manual describes how to use the Sockets API and OpenVMS system services to develop network applications.
VSI TCP/IP Services for OpenVMS SNMP Programming and Reference	This manual describes the Simple Network Management Protocol (SNMP) and the SNMP application programming interface (eSNMP). It describes the subagents provided with TCP/IP Services, utilities provided for managing subagents, and how to build your own subagents.
VSI TCP/IP Services for OpenVMS Tuning and Troubleshooting	This manual provides information about how to isolate the causes of network problems and how to

Manual	Contents
	tune the VSI TCP/IP Services software for the best performance.
VSI TCP/IP Services for OpenVMS Guide to IPv6	This manual describes the IPv6 environment, the roles of systems in this environment, the types and function of the different IPv6 addresses, and how to configure TCP/IP Services to access the IPv6 network.

For additional information about VSI OpenVMS products and services, see the website:

5. VSI Encourages Your Comments

You may send comments or suggestions regarding this manual or any VSI document by sending electronic mail to the following Internet address: <docinfo@vmssoftware.com>. Users who have OpenVMS support contracts through VSI can contact <support@vmssoftware.com> for help with this product.

6. OpenVMS Documentation

The full VSI OpenVMS documentation set can be found on the VMS Software Documentation webpage at https://docs.vmssoftware.com

7. Conventions

The name VSI TCP/IP Services means both:

- VSI TCP/IP Services for OpenVMS Alpha
- VSI TCP/IP Services for OpenVMS VAX

In addition, please note that all IP addresses are fictitious.

The following conventions are used in this manual. In addition,please note that all IP addresses are fictitious.

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 and then press and release another key or a pointing device button.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.) In the HTML version of this document, this convention appears as brackets, rather than a box.
…	A horizontal ellipsis in examples indicates one of the following possibilities:

	• Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In command format descriptions, parentheses indicate that you must enclose choices in parentheses if you specify more than one.
[ ]	In command format descriptions, brackets indicate optional choices. You can choose one or more items or no items. Do not type the brackets on the command line. However, you must include the brackets in the syntax for OpenVMS directory specifications and for a substring specification in an assignment statement.
|	In command format descriptions, vertical bars separate choices within brackets or braces. Within brackets, the choices are optional; within braces, at least one choice is required. Do not type the vertical bars on the command line.
{ }	In command format descriptions, braces indicate required choices; you must choose at least one of the items listed. Do not type the braces on the command line.
bold type	This typeface represents the introduction of a new term. It also represents the name of an argument, an attribute, or a reason.
italic type	Italic text indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*),in command lines (/PRODUCER=*name*), and in command parameters in text (where *dd* represents the predefined code for the device type).
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Monospace type`	Monospace type indicates code examples and interactive screen displays.

	This typeface indicates UNIX system output or user input,commands, options, files, directories, utilities, hosts, and users. In the C programming language, this typeface identifies the following elements: keywords, the names of independently compiled external functions and files, syntax summaries, and references to variables or identifiers introduced in an example.
-	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

Chapter 1. Troubleshooting Techniques and Tools

This chapter provides information that helps you identify symptoms, isolate problems, and take steps to resolve your network problem. This chapter also introduces the tools available to help you monitor and diagnose problems with your network software, devices, and interfaces.

1.1. Using Symptoms to Identify a Problem

The inability to reach remote hosts and networks is usually caused by one of the following:

- Physical connection failure
- Underlying transport failure (UDP, TCP, IP)
- Incorrectly configured routing, applications, or services such as BIND
- User error

1.2. Isolating Problems

The first step in problem isolation is to make sure that the VSI TCP/IP Services product is started. This may seem like an obvious step, but it is frequently overlooked because error messages may not indicate the product is disabled. (Instead, the messages returned may be “invalid host” or something similar.) You may not have stopped the product, but someone else may have. To check whether the product is running, enter the following command:

```
$ SHOW DEVICE BG

Device                  Device           Error
 Name                   Status           Count
BG0:                    Mounted              0
BG5:                    Mounted              0
BG6:                    Mounted              0
BG7:                    Mounted              0
BG8:                    Mounted              0
    .
    .
    .
```

If the command output shows only the BG0: device, then the product is stopped.

The second step is to reduce the problem to its basic components and to systematically identify what is and what is not working. Ask the following questions:

- Does the problem occur all the time or intermittently?
- Does it involve all hosts or is it limited to one host?
- Are there special load or configuration conditions under which you encounter a specific problem?
- Does the problem affect a single user? Multiple users? Your LAN?

The following steps can help you isolate your problem and determine a solution.

1. Check connectivity. (Section 1.2.1)
2. Check network interface parameters. (Section 1.2.2)
3. Check the IP address to Ethernet address translation tables. (Section 1.2.3)
4. Examine network statistics. (Section 1.2.4)
5. Monitor network traffic. (Section 1.2.5)
6. Check name server operation. (Section 1.2.7)
7. Check the route to a remote host. (Section 1.2.8)
8. Check the routes known to a gateway. (Section 1.2.9)
9. Check whether the network services have been enabled. (Section 1.2.10)
10. Look for application errors or interoperability issues.

Table 1.1 summarizes the tools you use to obtain information about network operations. The following sections describe each tool in detail.

Table 1.1. Diagnostic Tools

Diagnostic Tool	Function
`arp`	Controls and displays ARP tables.
`dig`	Sends domain name query packets to name servers.
`ifconfig`	Configures or displays network interface parameters, redefines an address for a particular interface, or sets options such as an alias list, broadcast address, or access filter. Use to detect incorrect IP addresses, subnet masks, and broadcast addresses.
`ndc`	Allows the name server administrator to send messages to a name server to start, stop, and restart BIND; to dump the BIND database; to check the status of the BIND process; and to change the tracing level.
`netstat`	Displays network statistics of sockets, data link counters, specified protocols or aliases, network interfaces, and a host's routing table.
`nslookup`	Provides the ability to directly query a name server and retrieve information. Use NSLOOKUP to determine whether your local name server is running correctly or to retrieve information from remote name servers.
`ping`	Indicates a host is reachable, and displays statistics about packet loss and delivery time.
`route`	Allows the user to manipulate the network routing tables manually.

Diagnostic Tool	Function
`sysconfig`	Displays and maintains the various network subsystem attributes.
`TCPTRACE`	Traces packets going in and out of the system. To run the trace utility, enter the DCL command TCPTRACE.
`traceroute`	Displays the route of an IP packet sent from the local host to a remote host.

To enter a command at the system prompt, first run the SYS$STARTUP:TCPIP $DEFINE_COMMANDS.COM procedure. This procedure defines each tool as a foreign command.

See Appendix A for complete reference information about these diagnostic tools.

1.2.1. Testing Connectivity Between Network Hosts

Use the `ping` command to test whether you can reach a remote host from your local system. The `ping` command sends an Internet Control Message Protocol (ICMP) echo request to the specified host name or host address. When received by a host, an ICMP reply is returned to the requester.

When using the `ping` command to isolate a problem, you should first test the `localhost` to verify that the system can communicate with itself. For example:

```
TCPIP> ping localhost
PING LOCALHOST (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=1 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=1 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0 ms

----LOCALHOST PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/0/1 ms
TCPIP>
TCPIP> ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=1 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=1 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0 ms

----127.0.0.1 PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/0/1 ms
```

The output from this `ping` command shows that the system is able to send a message down and then back up the protocol stack through the loopback address. The host address 127.0.0.1 and its associated host name, `localhost`, are the loopback address of the local host. This address was devised so that software could use common code to address local processes as well as remote processes. If the command output shows that it received a message for every message it transmitted, then you can be sure that the network software is up and running and that your system should able to communicate with remote systems.

If you do not receive output similar to that shown in the example, then one of the following conditions may exist:

- VSI TCP/IP Services may not be running.
- A definition for `localhost` is missing from the local database.

If the `ping` command for `localhost` does not respond correctly, try the `ping` command with the IP address `127.0.0.1`. If this command displays correct output, the TCPIP database is missing a definition for `localhost`.

If `localhost` returns the data correctly at this point, use the `ping` command to test another host on the same local network. If you are able to reach this host, then test remote hosts farther and farther away from the local host.

If the remote host does not respond to the request, the `ping` command displays the following message:

```
TCPIP> ping a7u1kt
ping: unknown host a7u1kt
%SYSTEM-F-UNREACHABLE, remote node is not currently reachable
```

If you used an IP address in the `ping` command, the output may be:

```
TCPIP> ping 10.10.22.1
PING 10.10.22.1 (10.10.22.1): 56 data bytes

----10.10.22.1 PING Statistics----
4 packets transmitted, 0 packets received, 100% packet loss
%SYSTEM-F-TIMEOUT, device timeout
```

These error messages could indicate that:

- There is no host with the specified host name.
- Using the host file or DNS/BIND, the system was not able to resolve the specified host name to an IP address.
- There is no host with that IP address.
- The host is down and not responding.

The following sample shows the `ping` statistics displayed:

```
TCPIP> ping chester
PING chester (16.20.208.53): 56 data bytes
64 bytes from 16.20.208.53: icmp_seq=0 ttl=64 time=0 ms
64 bytes from 16.20.208.53: icmp_seq=1 ttl=64 time=1 ms
64 bytes from 16.20.208.53: icmp_seq=2 ttl=64 time=0 ms
64 bytes from 16.20.208.53: icmp_seq=3 ttl=64 time=1 ms

----chester PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/0/1 ms
```

The `ping` command displays statistics on packets sent; packets received; the percentage of packets lost; and the minimum, average, and maximum round-trip packet times.

If you do not specify command options, the `ping` command displays the results of each ICMP request in sequence, the number of bytes received from the remote host, and the round-trip time on a per-request basis.

Use the output from the `ping` command to help determine the cause of direct and indirect routing problems such as host is unreachable, connection timed out, and network is unreachable.

This command helps you decide whether further testing is required and where. For example, if someone reports a problem connecting to a remote host, but `ping` shows packets traveling to the remote system and back, the problem probably resides in the upper (application) layer protocols (such as FTP, TELNET), or the user introduced the error to the application.

If the packets do not make the round trip, the problem probably resides in the lower layers, and perhaps indicates a misconfigured interface or other configuration or routing problems.

When preliminary testing indicates a problem in the lower layers, the next step is to test the network interfaces and routing. Use the `ifconfig`, `netstat`, and `arp` commands for these purposes (see Appendix A).

1.2.1.1. Using ping on a Multihomed Host

If you suspect one of the interfaces is down, you can test the interface by using:

- The `ping` command on another system to `ping` the IP address of the suspect interface. Because VSI TCP/IP Services sends messages via the first matching entry in the routing table, you might not be exercising the interface if you test on the local system.
- The `traceroute -s` *`src_addr`* command on the local system. This command uses the IP address specified with the `-s` flag in outgoing probe packets. If the command fails, the error message may indicate that the problem is with the interface.

1.2.2. Checking the Network Interface Parameters

Use the `ifconfig` command to check the configuration of a network interface. A common problem is a misconfigured subnet mask or incorrect IP address. Be sure to check the values of these parameters.

To display configuration information for all interfaces, enter the following command:

```
TCPIP> ifconfig -a
LO0: flags=100c89<UP,LOOPBACK,NOARP,MULTICAST,SIMPLEX,NOCHECKSUM>
     inet6 ::1
TN0: flags=80<NOARP>
WF0: flags=c43<UP,BROADCAST,RUNNING,MULTICAST,SIMPLEX>
    *inet 10.10.2.1 netmask ffffff00 broadcast 10.10.2.255 ipmtu 1500
     inet6 fe80::200:f8ff:febd:bc22
     inet6 3ffe:1200:4120:1000:200:f8ff:febd:bc22
```

For example, to display the configuration for interface WF0, enter the following command:

```
TCPIP> ifconfig WF0
```

The system displays the following information:

```
WF0: flags=c43<UP,BROADCAST,RUNNING,MULTICAST,SIMPLEX>
     inet 127.0.0.1 netmask ffff0000 broadcast 10.10.2.255 ipmtu 4470
     inet6 fe80::200:f8ff:febd:bc22
     inet6 3ffe:1200:4120:1000:200:f8ff:febd:bc22
```

The first line of this display shows the interface characteristics. The interface should be UP and RUNNING (exceptions to this are the LO0 and TN0 interfaces). The pertinent fields in this display show the interface's IP address, the subnet mask, the broadcast mask, and the maximum transmit unit.

You can also obtain information using the following commands:

```
TCPIP> SHOW INTERFACE
                                                      Packets
Interface   IP_Addr        Network mask          Receive          Send    MTU
 LO0        127.0.0.1      255.0.0.0                 137           137   4096
 WE0        10.10.2.1      255.255.255.0            5089          4191   1500

TCPIP> SHOW CONFIGURATION INTERFACE
 Interface: LO0
   IP_Addr: 127.0.0.1         NETWRK: 255.0.0.0         BRDCST:
 Interface: WE0
   IP_Addr: 10.10.2.1         NETWRK: 255.255.255.0     BRDCST: 10.10.2.255
```

If you are not familiar with IP addressing and the concepts of subnet and broadcast masks, review the information in *VSI TCP/IP Services for OpenVMS Management* before proceeding with troubleshooting tasks.

1.2.3. Displaying and Modifying the Internet-to-Ethernet Translation Tables

Use the `arp` utility or the SHOW ARP management command to check the IP address to Ethernet address translation entries in the Address Resolution Protocol (ARP) table. This is useful if you think incorrect entries are being added to the ARP table. For example, if you enter a command and an unexpected host responds, you may have two systems defined with the same IP address in the ARP table.

To display entries in the ARP table, enter the following command:

```
$ TCPIP SHOW ARP
Cnt  Flags    Timer Host                    Phys Addr
  1: UCS        451 160.20.0.10             08-00-2b-39-c7-ac
  2: UC           0 160.20.0.100            aa-00-04-00-8d-13
  3: UC           3 160.20.0.173            00-00-f8-45-a0-b4
  4: UC          14 160.20.32.94            00-00-f8-00-f7-41
  5: UC          50 160.20.64.69            00-d0-b7-19-78-a4
  6: UCS          9 160.20.64.132           00-50-8b-72-7f-ff
  7: UCS        150 160.20.80.124           00-50-8b-4d-91-b3
```

The following TCP/IP Services management commands allow you to configure the hardware addresses for remote IP addresses:

- SET ARP allows you to add the hardware address to the ARP table.
- SET PROTOCOL ARP allows you specify the time interval for ARP to hold information in its cache. Use the SET CONFIGURATION PROTOCOL ARP command to set this information in the permanent configuration database.

For more information about these commands, refer to the *VSI TCP/IP Services for OpenVMS Management Command Quick Reference Card* manual.

For information about using the `arp` utility, refer to Appendix A.

1.2.4. Examining Network Statistics

Use the `netstat` utility or the SHOW INTERFACE command to check interface and protocol statistics, per-connection status, and memory buffer use. Look for bad checksums, excessive retransmissions, dropped packets, out-of-order packets, and lost-carrier errors.

For example:

```
TCPIP> netstat -i
Name  Mtu   Network  Address               Ipkts Ierrs    Opkts Oerrs  Coll
TN0*  1280  Link     Link#2                    0     0        0     0     0
WF0   4470  Link     00:00:f8:cd:1e:e4  48855499     0  2035244     0     0
WF0   4470  16.20    ucxaxp             48855499     0  2035244     0     0
LO0   4096  Link     Link#1               165084     0   165084     0     0
LO0   4096  127      LOCALHOST            165084     0   165084     0     0
```

Some problems to look for are:

- If the output of the `netstat -i` command shows an excessive amounts of input errors (Ierrs), output errors (Oerrs), or collisions (Coll), this may indicate a network problem; for example, cables may not be connected properly or the Ethernet may be saturated.

- Memory problems

  The `netstat -m` command shows statistics for network-related data structures. Use this command to determine if the network is using an excessive amount of memory in proportion to the number of sockets in use.

  The `netstat -an` displays allocated sockets (each socket results in a network connection).

- Network connections

  Use `netstat -an` to check allocated sockets (each socket results in a network connection).

The following example shows the output of the `netstat -m` command:

```
TCPIP> netstat -m
1328 mbufs in use:
        2 mbufs allocated to data
        2 mbufs allocated to ancillary data
        139 mbufs allocated to socket structures !❶
        244 mbufs allocated to protocol control blocks "❷
        442 mbufs allocated to routing table entries
        2 mbufs allocated to socket names and addresses
        26 mbufs allocated to interface addresses
        1 mbufs allocated to ip multicast options
        2 mbufs allocated to ip multicast addresses
        5 mbufs allocated to interface multicast addresses
        48 mbufs allocated to NFS request header
        1 mbufs allocated to vnode struct
        32 mbufs allocated to kern credential data structure
```

```
2 mbufs allocated to mbuf extra-data protocol message
58 mbufs allocated to assorted NFS structures
6 mbufs allocated to network interface structure
2 mbufs allocated to netisr thread queues
5 mbufs allocated to Inet PCB queues
1 mbufs allocated to rad specific structures
1 mbufs allocated to OpenVMS Cluster Alias table
37 mbufs allocated to OpenVMS Kernel VCI structure
12 mbufs allocated to OpenVMS TCPIP Timer structure
3 mbufs allocated to OpenVMS LAN VCI VCIB structure
3 mbufs allocated to OpenVMS LAN MCAST_REQ structure
12 mbufs allocated to OpenVMS SELECT structure
1 mbufs allocated to OpenVMS ACP Filter Buffer
1 mbufs allocated to OpenVMS ACP AQB
1 mbufs allocated to OpenVMS ACP INETCB
2 mbufs allocated to OpenVMS Driver requested REQCB
26 mbufs allocated to OpenVMS ACP allocated SERV Structure
37 mbufs allocated to OpenVMS VCI context block
1 mbufs allocated to OpenVMS ACP IPCACHE Structure
57 mbufs allocated to OpenVMS PROXY correlation records
66 mbufs allocated to OpenVMS PROXY host records
14 mbufs allocated to OpenVMS PROXY local user records
17 mbufs allocated to OpenVMS PROXY remote user records
17 mbufs allocated to OpenVMS Unix emulation stack (NFS, et al)
```

❶ This line indicates there are 24 sockets in use (1 mbuf allocated for each socket).

❷ There are two protocol control blocks allocated for each TCP socket and one protocol control block for each UDP socket. The 35 mbuf listed is a mix of PCBs allocated for TCP and UDP sockets. Output from the TCPIP SHOW DEVICE_SOCKET command will tell you how many TCP and UDP sockets are allocated.

By comparing the information output from the `netstat -m` and the TCPIP command SHOW DEVICE_SOCKET, you can estimate whether the system is using an excessive amount of memory for the number of allocated sockets.

If you sense that TCP/IP Services is using an excessive amount of memory for the number of sockets, there may be a memory leak. Capture the output from the `netstat -m` and the TCPIP SHOW DEVICE_SOCKET commands and save for documenting the condition.

Table 1.2 shows variations of the netstat command that can reveal network problems.

Table 1.2. `netstat` Commands

Command	Purpose
`netstat -p ip`	Checks for bad checksums, length problems, excessive redirects, and packets lost because of resource problems.
`netstat -p tcp`	Checks for retransmissions, out of order packets, and bad checksums.
`netstat -p udp`	Looks for bad checksums and full sockets.
`netstat -rs`	Obtains routing statistics.
`netstat -s`	Simultaneously displays statistics related to the IP, ICMP, TCP, and UDP protocol layers.
`netstat -is`	Checks for network device driver errors.

For more information about `netstat`, see Appendix A

1.2.5. Monitoring Network Traffic

You can use either of the following to monitor network traffic:

- The TCPTRACE command, described in Section 1.2.5.1
- The `tcpdump` utility, described in Section 1.2.5.2

1.2.5.1. Using TCPTRACE

The trace utility (TCPTRACE) is a tool you can use to trace packets going in and out of the system. To run the trace utility, enter the DCL command TCPTRACE. Use the qualifiers listed in the command reference section to customize tracing for your particular problem. For example:

```
$ TCPTRACE HOST1 /FULL /PORT=REMOTE=21

$ TCPTRACE HOST2 /PORT=(LOCAL=23, REMOTE=1056) /FULL /PACKETS=30 /
OUTPUT=TELNET_TRACE.TXT
```

The following sample is a TCPTRACE display:

```
TCPIP INTERnet trace RCV packet seq # = 1 at 23-OCT-1998 15:19:33.29

IP Version = 4,  IHL = 5,  TOS = 00,   Total Length = 217 = ^x00D9
IP Identifier  = ^x0065,  Flags (0=0,DF=0,MF=0),
      Fragment Offset = 0 = ^x0000,   Calculated Offset = 0 = ^x0000
IP TTL = 32 = ^x20,  Protocol = 17 = ^x11,  Header Checksum = ^x8F6C
IP Source Address      = 16.20.168.93
IP Destination Address = 16.20.255.255

UDP Source Port = 138,   UDP Destination Port = 138
UDP Header and Datagram Length = 197 = ^x00C5,   Checksum = ^x0E77

5DA81410   8F6C1120   00000065   D9000045    0000    E...awe.....l....]
         | 0E77C500   8A008A00 | FFFF1410    0010    ..........w.
```

For more information about using TCPTRACE, see Appendix A.

1.2.5.2. Using TCPDUMP

The OpenVMS `tcpdump` utility can trace natively on Ethernet or can format traces taken on another host. It communicates with the TCP/IP kernel in copy-all mode so it only can trace packets received or transmitted by the TCP/IP kernel.

The trace can be taken interactively and ended with Ctrl/C, or continue until a packet count has been reached (specified using `-c` *`count`*). The `tcpdump` utility displays a summary line indicating the number of packets traced and the number of packets discarded by the kernel.

To use `tcpdump`, no special TCP/IP Services configuration is required. The process using `tcpdump` must have OPER, PSWAPM, and CMKRNL privileges.

The format and options are described in Appendix A. This section describes:

- How to build expressions (Section 1.2.5.2.1)

- How to analyze output from `tcpdump` (Section 1.2.5.3)
- Restrictions on using OpenVMS `tcpdump` (Section 1.2.5.4)
- How to reduce discarded packets (Section 1.2.5.5)

1.2.5.2.1. Building Expressions

The expression is used to select the packets to dump. If no expression is given, all packets on the network are dumped. Otherwise, only packets for which *expression* is TRUE are dumped.

The expression consists of one or more primitives. Primitives usually consist of an identifier (name or number) preceded by one or more of the keywords described in Table 1.3

Table 1.3. `tcpdump` Keywords

Keyword	Definition

type	Defines the object to which the ID name or number refers. The following types are allowed:
	• `host`
	• `net`
	• `port`
	For example:
	`host foo`
	`net 128.3`
	`port 20`
	If no *type* keyword is specified, host is the default.

dir	Specifies a particular transfer direction to or from *id*. The following directions are allowed:
	• `src`
	• `dst`
	• `src or dst`
	• `src and dst`
	For example:
	`src foo`
	`dst net 128.3`
	`src or dst port 20`
	`src and dst port 123`
	If no *dir* keyword is specified, src or dst is the default.

proto	Restricts the match to a particular protocol. The following protocols are allowed: • `ether`

Keyword	Definition

	• `ip`
	• `ipv6`
	• `icmpv6`
	• `arp`
	• `tcp`
	• `udp`
	For example:
	`ether src foo`
	`arp net 128.3`
	`tcp port 21`
	If no *proto* keyword is specified, all protocols consistent with the *type* are assumed. For example, `src foo` means:
	`(ip or arp or rarp) src foo`
	(Note that the latter is not valid syntax.)
	net bar means:
	`(ip or arp) net bar`
	port 53 means:
	`(tcp or udp) port 53`

1.2.5.2.2. Keywords for Filtering Traces from non-OpenVMS Systems

To filter a trace taken on another platform, the following keywords are available:

- `ether`
- `fddi`
- `ip`
- `ipv6`
- `icmpv6`
- `arp`
- `rarp`
- `decnet`
- `lat`
- `moprc`

- `mopdl`
- `tcp`
- `udp`

1.2.5.2.3. Primitive Keywords

Primitive keywords include:

- `gateway`
- `broadcast`
- `less`
- `greater`
- `Arithmetic expressions`

More complex filter expressions are formed by using the words and, or, and not to combine primitives. For example:

```
host foo and not port ftp and not port ftp-data
```

To minimize keystrokes, identical keyword lists can be omitted. For example, the following two lines are treated the same:

```
tcp dst port ftp or ftp-data or domain
tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain
```

1.2.5.2.4. Primitive Expressions

The following list describes the results of using some primitive expressions.

- `dst host` *`host`*

  True if the IP destination field of the packet is *host*, which may be either an address or a name.

- `src host` *`host`*

  True if the IP source field of the packet is *host*.

- `host` *`host`*

  True if either the IP source or destination of the packet is *host*. The following keywords can precede any of these host expressions:

  - `ip`
  - `arp`
  - `rarp`

  The following examples are equivalent:

  `ip host` *`host`*

```
ether proto ip and host host
```

If *host* is a name with multiple IP addresses, each address is checked for a match.

- `ether dst ehost`

  True if the Ethernet destination address is *ehost*.

- `ether src ehost`

  True if the Ethernet source address is *ehost*.

- `ether host ehost`

  True if either the Ethernet source or destination address is *ehost*.

- `gateway host`

  True if the packet used *host* as a gateway. That is, the Ethernet source or destination address was *host* but neither the IP source nor the IP destination was *host*.

  This expression is equivalent to the following:

  ```
  ether host ehost and not host host
  ```

You can use either names or numbers for *host* and *ehost*.

- `dst net net`

  (IPv4 networks only) True if the IP destination address of the packet has a network number of *net*, which may be either an address or a name.

- `src net net`

  (IPv4 networks only) True if the IP source address of the packet has a network number of *net*.

- `net net`

  (IPv4 networks only) True if either the IP source or destination address of the packet has a network number of *net*.

- `dst port port`

  True if the packet is IP/TCP or IP/UDP and has a destination port value of *port*.

- `src port port`

  True if the packet has a source port value of *port*.

- `port port`

  True if either the source or destination port of the packet is *port*. The following keywords can precede any of these port expressions:

  - `tcp`
  - `udp`

For example, the following example matches only TCP packets:

```
tcp src port port
```

- `less length`

  True if the packet has a length less than or equal to *length*. The following example is equivalent:

  ```
  len <= length
  ```

- `greater length`

  True if the packet has a length greater than or equal to *length*. The following example is equivalent:

  ```
  len >= length
  ```

- `ip proto protocol`

  True if the packet is an IP packet of protocol type *protocol*. The protocol can be a number, or one of the following names:

  - `ipv6`
  - `icmp`
  - `icmpv6`
  - `udp`
  - `nd`
  - `tcp`

- ether broadcast

  True if the packet is an Ethernet broadcast packet. The ether keyword is optional.

- ip broadcast

  (IPv4 networks only) True if the packet is an IP broadcast packet. It checks for both the all-zeroes and all-ones broadcast conventions, and looks up the local subnet mask.

- ether multicast

  True if the packet is an Ethernet multicast packet. The ether keyword is optional. This is shorthand for:

  ```
  ether[0] & 1 != 0
  ```

- `ip multicast`

  (IPv4 networks only) True if the packet is an IPv4 multicast packet.

- `ether proto protocol`

  True if the packet is of protocol type ether. The protocol argument can be a number or a name, such as `ip`, `ipv6`, and `arp`.

Only Ethernet is supported with protocols ip, ipv6 and arp for native tracing. If reading a trace created on another platform, tcpdump will be able to filter and format it correctly.

- `expr relop expr`

  True if the relation holds, where *relop* is one of the following:

  - `>`
  - `<`
  - `>=`
  - `<=`
  - `=`
  - `,`
  - `!=`

  expr is an arithmetic expression composed of integer constants (expressed in standard C syntax), the normal binary operators [+, -, *, /, &, |], a length operator, and special packet data accessors.

1.2.5.2.5. Accessing Data Inside Packets

To access data inside the packet, use the following syntax:

`proto [expr : size]`

The following list describes the variables.

- The *proto* variable is one of the following:
  - `ether`
  - `fddi`
  - `ip`
  - `arp`
  - `rarp`
  - `tcp`
  - `udp`
  - `icmp`

  The *proto* variable indicates the protocol layer for the index operation.
- The byte offset, relative to the indicated protocol layer, is specified by *expr*.
- The *size* variable is optional and indicates the number of bytes in the field of interest; it can be:
  - `one`

- `two`
- `or`
- `four`

The default size is `one`.

For example:

- `ether[0]&1!=0` detects all multicast traffic.
- `ip[0] & 0xf != 5` detects all IP packets with options.
- `ip[2:2] & 0x1fff = 0` detects only unfragmented datagrams and fragment zero of fragmented datagrams. This check is implicitly applied to the TCP and UDP index operations. For instance, `tcp[0]` always means the first byte of the TCP header, and never means the first byte of an intervening fragment.

1.2.5.2.6. Combining Keywords

Keywords can be combined using:

- A parenthesized group of primitives and operators
- Negation (`!` or `not`)
- Concatenation (`and`)
- Alternation (`or`)

Negation has highest precedence. Alternation and concatenation have equal precedence and associate left to right. Note that explicit and tokens (not juxtaposition) are required for concatenation.

If an identifier is given without a keyword, the most recent keyword is assumed. For example, the following two examples are equivalent:

```
not host vs and ace
not host vs and host ace
```

However, the following example is not equivalent to the previous two:

```
not ( host vs or ace )
```

Expression arguments can be passed to `tcpdump` as either a single argument or as multiple arguments, whichever is more convenient. Multiple arguments are concatenated with spaces before being parsed.

1.2.5.3. Analyzing Output

The output of the `tcpdump` utility is protocol dependent. The following sections describe the formats and provide examples.

1.2.5.3.1. Link Level Headers

The `-e` option is used to display the link level header. On Ethernet networks, the source and destination addresses, protocol, and packet length are displayed.

Only Ethernet frame types are supported with `tcpdump`.

1.2.5.3.2. ARP Packets

ARP output shows the type of request and its arguments. The format is intended to be self explanatory. The following example is taken from the start of an RLOGIN session from host `rtsg` to host `csam`:

```
arp who-has csam tell rtsg
arp reply csam is-at CSAM
```

The first line indicates that host rtsg sent an ARP packet asking for the Ethernet address of Internet host csam. Host csam replies with its Ethernet address (in this example, Ethernet addresses are uppercase and Internet addresses in lowercase).

This is equivalent to:

```
arp who-has 128.3.254.6 tell 128.3.254.68
arp reply 128.3.254.6 is-at 02:07:01:00:01:c4
```

If you issue the `tcpdump -e` command, the first packet is explicitly a broadcast packet and the second is a point-to-point packet:

```
RTSG Broadcast 0806  64: arp who-has csam tell rtsg
CSAM RTSG 0806  64: arp reply csam is-at CSAM
```

For the first packet, the Ethernet source address is RTSG, the destination is the broadcast address, the type field contain hex 0806 (type ETHER_ARP) and the total length is 64 bytes.

1.2.5.3.3. TCP Packets

The following description assumes familiarity with the TCP protocol described in RFC 793.

The general format of a TCP protocol line is:

```
src > dst: flags data-seqno ack window options
```

The fields represent the following:

- *src*

  The source IP addresses and ports.

- *dst*

  The destination IP addresses and ports.

- *flags*

  The sum combination of S (SYN), F (FIN), P (PUSH), or R (RST) or a single period (.) for no flags.

- *data-seqno*

  The portion of sequence space covered by the data in this packet (see the example below).

- *ack*

  The sequence number of the next data expected from the other direction on this connection.

- *window*

  The number of bytes of receive buffer space available from the other direction on this connection.

- `urgent`

  Indicates there is urgent data in the packet.

- *options*

  The TCP options enclosed in angle brackets. For example:

  `<mss 1024>`

The `src`, `dst`, and `flags` fields are always present. The other fields depend on the contents of the packet's TCP protocol header and are output only if appropriate.

Examples

The following example shows the opening portion of an RLOGIN session from host rtsg to host csam:

```
rtsg.1023 > csam.login: S 768512:768512(0) win 4096 <mss 1024>
csam.login > rtsg.1023: S 947648:947648(0) ack 768513 win 4096 <mss 1024>
rtsg.1023 > csam.login: . ack 1 win 4096
rtsg.1023 > csam.login: P 1:2(1) ack 1 win 4096
csam.login > rtsg.1023: . ack 2 win 4096
rtsg.1023 > csam.login: P 2:21(19) ack 1 win 4096
csam.login > rtsg.1023: P 1:2(1) ack 21 win 4077
csam.login > rtsg.1023: P 2:3(1) ack 21 win 4077 urg 1
csam.login > rtsg.1023: P 3:4(1) ack 21 win 4077 urg 1
```

The example shows the following sequence of communication:

- The first line indicates that TCP port 1023 on system rtsg sent a packet to port login on host csam. The S indicates that the SYN flag was set. The packet sequence number was 768512 and it contained no data. (The notation is *first:last(nbytes)*, which means sequence numbers *first* up to but not including *last*, which is *nbytes* bytes of user data.) There was no piggy-backed ack, the available receive window was 4096 bytes and there was a `max-segment-size` option requesting an mss of 1024 bytes.

- Host csam replies with a similar packet except that it includes a piggy-backed ack for the SYN sent by rtsg.

- Host rtsg then sends an ack reply to the SYN sent by csam. The period (.) means no flags were set. The packet contained no data, so there is no data sequence number. Note that the ack sequence number is a small integer (1). The first time tcpdump sees a TCP conversation, it displays the sequence number from the packet.

- On subsequent packets of the conversation, the difference between the current packet's sequence number and this initial sequence number is displayed. Thus, sequence numbers after the first can be interpreted as relative byte positions in the conversation's data stream (with the first data byte each direction being 1). The -S option overrides this feature, causing the original sequence numbers to be output.

- The sixth line indicates that host rtsg sends host csam 19 bytes of data (bytes 2 through 20 in the rtsg-to-csam side of the conversation). The PUSH flag is set in the packet.

- The seventh line indicates that host csam has received data sent by host rtsg up to but not including byte 21. Most of this data is apparently sitting in the socket buffer because the receive window on host csam is 19 bytes smaller. Host csam also sends one byte of data to host rtsg in this packet.
- The eighth and ninth lines show that host csam sends two bytes of urgent, pushed data to rtsg.

1.2.5.3.4. UDP Packets

The UDP format is illustrated by the following RWHO packet:

```
actinide.who > broadcast.who: udp 84
```

This line of output indicates that port who on host actinide sent a UDP datagram to port who on host broadcast, the Internet broadcast address. The packet contained 84 bytes of user data.

Some UDP services are recognized (from the source or destination port number) and the higher level protocol information displayed, specifically Domain Name service requests (RFC 1034 and RFC 1035) and Sun RPC calls (RFC 1050) to NFS.

1.2.5.3.5. UDP Name Server Requests

The following description assumes familiarity with the Domain Service protocol described in RFC 1035.

Name server requests are formatted as follows:

```
src > dst: id op? flags qtype qclass name (len)
```

For example:

```
h2opolo.1538 > helios.domain: 3+ A? ucbvax.berkeley.edu. (37)
```

Host h2opolo queried the domain server on host helios for an address record (*qtype*=A) associated with the name ucbvax.berkeley.edu. The query ID was 3. The plus sign (+) indicates the recursion desired flag was set. The query length was 37 bytes, not including the UDP and IP protocol headers. The query operation was the normal one, Query, so the *op* field was omitted. If the *op* field had been anything else, it would have been displayed between the 3 and the plus sign (+). Similarly, the *qclass* was the normal one, C_IN, and omitted. Any other *qclass* would have been displayed immediately after the A.

The following anomalies are checked and may result in extra fields enclosed in square brackets:

- If a query contains an answer, name server, or authority section, ancount, nscount, or arcount are displayed as [*n*a], [*n*n] or [*n*au], where *n* is the appropriate count.
- If any of the response bits are set (AA, RA or rcode) or any of the "must be zero" bits are set in bytes 2 and 3, [b2&3=*x*]) is displayed, where *x* is the hexadecimal value of header bytes 2 and 3.

1.2.5.3.6. UDP Name Server Responses

Name server responses are formatted as follows:

```
src > dst:  id op rcode flags a|n|au type class data (len)
```

For example:

```
helios.domain > h2opolo.1538: 3 3/3/7 A 128.32.137.3 (273)
helios.domain > h2opolo.1537: 2 NXDomain* 0/1/0 (97)
```

In the first example, host helios responds to query ID 3 from host h2opolo with 3 answer records, 3 name server records, and 7 authority records. The first answer record is type A (address) and its data is Internet address 128.32.137.3. The total size of the response is 273 bytes, excluding UDP and IP headers. The *op* (Query) and response code (NoError) are omitted, as is the class (C_IN) of the A record.

In the second example, host helios responds to query 2 with a response code of nonexistent domain (NXDomain) with no answers, one name server and no authority records. The asterisk (*) indicates that the authoritative answer bit is set. Since there are no answers, no type, class, or data are displayed.

Other flag characters that might appear are the minus sign (-) (recursion available, RA, not set) and vertical bar (|) (truncated message, TC, set). If the "question" section does not contain exactly one entry, [nq] is displayed.

Note that name server requests and responses tend to be large, and the default value of *snaplen*, 96 bytes, may not capture enough of the packet to print. Use the -s option to increase the *snaplen* if you need to seriously investigate name server traffic.

1.2.5.3.7. Sun RPC Requests and Replies

Sun RPC (RFC 1057) is decoded as described in Table 1.4, as are several of the protocols that use Sun RPC.

Table 1.4. SUN RPC Requests

Name	Users	Description
PORTMAP	libc.a, portmap	Maps RPC program numbers to TCP/UDP ports.
MOUNT	mount, mountd	Maps file names to NFS file handles.
NLM	rpc.lockd	NFS remote file locking.
STAT	rpc.statd, rpc.lockd	Remote status monitor.
YP	libc.a, ypserv	Network Information Services.
YPBIND	ypbind, ypset	NIS domain manipulation.
NFS	UNIX	Network File System.

Requests sent using TCP must start at the beginning of a packet to be decoded. Normally they are; however, applications that have multiple requests outstanding (for example, NFS) may not always do this.

Replies can only be decoded if the request was found and only if they start a packet.

The form of an RPC request and reply is as follows:

```
src.xid > dst.prot-vn: len call op args
src.xid > dst.prot-vn: len reply op results
```

For example, NFS mounting a file system generates:

```
clnt.312dbc68 > svc.pmap-v2: 56 call getport prog "nfs"  V3  prot UDP po0
svc.312dbc68 > clnt.pmap-v2: 28 reply getport 2049
clnt.312deff8 > svc.pmap-v2: 56 call getport prog "mount"  V3  prot UDP 0
svc.312deff8 > clnt.pmap-v2: 28 reply getport 1034
clnt.312deff8 > svc.mount-v3: 124 call mount "/build"
svc.312deff8 > clnt.mount-v3: 68 reply mount  OSF/1 fh 8,3079/1.2
clnt.907312 > svc.nfs-v3: 148 call getattr OSF/1 fh 8,3079/1.2
svc.907312 > clnt.nfs-v3: 112 reply getattr {dir size 1024 mtime ... }
```

The UDP or TCP protocol information is not displayed. This is generally not important for UDP; however, it can be important for TCP. If the `-m` and `-v` options are in effect, both RPC and TCP decoding are done. For example, the UNIX command `showmount -e srv` generates information such as the following:

```
clnt.3123f473 > svc.pmap-v2: 56 call getport prog "mount"  V1  prot TCP 0
                 (ttl 29, id 19672)
svc.3123f473 > clnt.pmap-v2: 28 reply getport 892
                 (ttl 30, id 31644)
clnt.1032 > svc.892: S 25280000:25280000(0) win 32768 <mss 1460,nop,wsca>
                 (DF) (ttl 59, id 19674)
svc.892 > clnt.1032: S 483136000:483136000(0) ack 25280001 win 33580
                 <mss 1460,nop,wscale 0> (ttl 60, id 31645)
clnt.1032 > svc.892: . ack 1 win 33580 (DF) (ttl 59, id 19675)
clnt.2f221c23 > svc.mount-v1: 40 call return export list
TCP: clnt.1032 > svc.892: P 1:45(44) ack 1 win 33580 (DF) (ttl 59, id 19)
svc.2f221c23 > clnt.mount-v1: 184 reply export
        "/usr": "client" "clnt"
        "/build":
                ...
TCP: svc.892 > clnt.1032: P 1:189(188) ack 45 win 33580 (ttl 60, id 3164)
clnt.1032 > svc.892: F 45:45(0) ack 189 win 33580 (DF) (ttl 59, id 19679)
svc.892 > clnt.1032: . ack 46 win 33580 (ttl 60, id 31649)
svc.892 > clnt.1032: F 189:189(0) ack 46 win 33580 (ttl 60, id 31650)
clnt.1032 > svc.892: . ack 190 win 33580 (DF) (ttl 59, id 19681)
```

The following is another NFS sample:

```
sushi.6709 > wrl.nfs-v2: 112 call readlink fh 21,24/10.731657119
wrl.6709 > sushi.nfs-v2: 40 reply readlink "../var"
sushi.201b > wrl.nfs-v2: 144 call lookup fh 9,74/4096.6878 "xcolors"
wrl.201b > sushi.nfs-v2: 128 reply lookup fh 9,74/4134.3150
```

The example shows the following sequence of communication:

- In the first line, host `sushi` sends a transaction with ID 6709 to host `wrl`. (The number following the `src` host is a transaction ID, not the source port.) The request was 112 bytes, excluding the UDP and IP headers. The operation was a readlink (read symbolic link) on file handle (`fh`) 21,24/10.731657119. (In some cases, the file handle can be interpreted as a major and minor device number pair, followed by the inode number and generation number.) Host `wrl` replies with the contents of the link.
- In the third line, host `sushi` asks host `wrl` to look up the name `xcolors` in directory file 9,74/4096.6878.
- The data displayed depends on the operation type. The format is intended to be self explanatory if read in conjunction with a protocol specification rpcgen `.x` file.

If the `-v` (verbose) option is given, additional information is displayed.

If the `-v` option is given more than once, more details may be displayed.

Note that RPC requests are very large and much of the detail is not displayed. Property list information may also be obtained using `tcpdump`. For example:

```
node1.abc.com.da31fba5 > node2.abc.com.proplist-v3: \
     276 call proproc3_get OSF/1 fh 8,18434/1.4 mask:-1 11 entries

node2.abc.com.da31fba5 > node1.abc.com.proplist-v3: \
     296 reply proproc3_get status OK 368 bytes 11 entries
```

For property list calls, you can request the mask value and the number of property list entries. Property list replies return the status, the number of bytes in the property list and the number of entries in property list.

Note that NFS requests are very large and much of the detail is not displayed unless the value of *snaplen* is increased. Use `-s` 192 to watch RPC traffic.

RPC reply packets do not explicitly identify the RPC operation. Instead, tcpdump keeps track of recent requests, and matches them to the replies using the transaction ID. If a reply does not closely follow the corresponding request, it might not be parsable.

1.2.5.3.8. IP Fragmentation

Fragmented Internet datagrams are printed as follows:

```
(frag id:size@offset+)
(frag id:size@offset)
```

The first line indicates there are more fragments. The second indicates this is the last fragment.

The following list explains the fields:

- *id* is the fragment ID.
- *size* is the fragment size (in bytes), excluding the IP header.
- *offset* is the fragment's offset (in bytes) in the original datagram.

The fragment information is output for each fragment. The first fragment contains the higher level protocol header and the fragment information is displayed after the protocol information. Fragments after the first contain no higher level protocol header and the fragment information is printed after the source and destination addresses. The following example shows part of an FTP session from arizona.edu to lbl-rtsg.arpa over a CSNET connection that does not appear to handle 576 byte datagrams:

```
arizona.ftp-data > rtsg.1170: . 1024:1332(308) ack 1 win 4096
(frag 595a:328@0+)
arizona > rtsg: (frag 595a:204@328)
rtsg.1170 > arizona.ftp-data: . ack 1536 win 2560
```

Note the following:

- Addresses in the second line do not include port numbers. This is because the TCP protocol information is in the first fragment and `tcpdump` does not know what the port or sequence numbers are when it displays the later fragments.

- TCP sequence information in the first line is displayed as if there were 308 bytes of user data; however, there are 512 bytes (308 in the first fragment and 204 in the second). If you are looking for holes in the sequence space or trying to match up acknowledgements with packets, this can be misleading.

  A packet with the IP "do not fragment" flag is marked with a trailing (DF).

1.2.5.3.9. Timestamps

By default, all output lines are preceded by a timestamp. The timestamp is the current clock time in the following form:

hh:mm:ss.frac

It is as accurate as the kernel's clock. The timestamp reflects the time the kernel first saw the packet. No attempt is made to account for the time difference between when the Ethernet interface removed the packet from the wire and when the kernel serviced the new packet interrupt.

1.2.5.4. Restrictions

The following restrictions apply to using `tcpdump` on OpenVMS:

- Copy-all mode is on by default on OpenVMS.
- Promiscuous mode is not available, so tracing must be issued on either the source or destination host.
- Only Ethernet native tracing on is supported on OpenVMS.
- Only one user may trace at a time on OpenVMS using either `tcpdump` or `tcptrace`.
- Name server inverse queries are not dumped correctly: The (empty) question section is displayed rather than real query in the answer section.
- A packet trace that crosses a daylight saving time change produces skewed time stamps (the time change is ignored).

1.2.5.5. Reducing Discarded Packets

When packets are copied by the TCP/IP kernel, it places them into a ring buffer that is emptied by `tcpdump`. If packets are received fast enough, the ring will fill up and the TCP/IP kernel discards (drops) packets until tcpdump has caught up. Because `tcpdump` has not seen these dropped packets, it cannot tell whether they were relevant to the requested trace.

If the option `-B` is used, `tcpdump` indicates when the drops occur by issuing a BUFFERSFULL error. This can be useful if the drops occur outside the sequence being analyzed.

There are several methods for reducing the number of packet drops:

- Specify a more detailed filter in the `tcpdump` command. Trace to a file instead of SYS$OUTPUT using `-w` *`filename`*. For best results, use a disk with little activity or a RAM disk.
- Increase the number of buffers in the ring using `-b` *`buffers`*. The default for Alpha systems is 400. For VAX systems, the default is 50. The processes working set quota (WSQUOTA) may need to be increased for larger numbers than the default.

- Increase the default process priority of the process that issued the tcpdump `command`.

1.2.6. Monitoring Socket Activity

TCP/IP Services provides a call tracing facility that can be used to help characterize and debug the use of the sockets API for many applications.

To enable tracing, define the TCPIP$SOCKET_TRACE logical name. The logical name accepts the following arguments:

- 1 or 0

  Specify 1 to enable socket tracing, or 0 to disable socket tracing. When the logical name is set to 1, the output from the trace is displayed interactively. For example:

  ```
  $ DEFINE TCPIP$SOCKET_TRACE 1
  ```

- Log file name

  Specify the name of the log file for storing the tracing information. For example:

  ```
  $ DEFINE TCPIP$SOCKET_TRACE SYS$LOGIN:TCPIP$SOCKET_TRACE.LOG
  ```

- Location for process-specific log files

  Specify a directory for storing the log files. Each log file name reflects the name of the process that is being traced. For example:

  ```
  $ DEFINE /SYSTEM TCPIP$SOCKET_TRACE SYS$SYSDEVICE:[LOGFILES]
  ```

The following example shows a sample tracing:

```
23:35:47.48 +socket family: 2, type: 1, proto: 0
23:35:47.48 -socket chan: 0xf0, st: 0x1, iosb: 0x1 0
23:35:47.48 *setsockopt sock: 0xf0, lev: 0xffff, opt: 0x4, val: 1, len: 4
23:35:47.49 *bind44 socket: 0xf0, st: 0x1, iosb: 0x1 0
23:35:47.50 *listen sock: 0xf0, backlog: 5
23:35:47.51 +accept44 chan: 0xf0
23:35:54.04 -accept44 rtchan: 0x100, st: 0x1, iosb: 0x1 0
23:35:54.04 *getpeername44 sock: 0x100
23:35:54.04 +send_64 sock: 0x100, addr: 0x7AEF7A00, len: 28, flags: 0x0
23:35:54.04 -send_64 st: 0x1, iosb: 0x1 28
23:35:54.04 *shutdown sock: 0x100, how: 2
23:35:54.05 *close sock: 0x100, st: 0x1
23:35:54.05 *close sock: 0xf0, st: 0x1
```

In this example, you can see the application opening a socket, setting socket options, binding, listening, accepting, sending data, and so forth.

Lines beginning with a plus sign (+) indicate that the relevent routine is being entered. There is usually a line beginning with a minus sign (-) soon after, when the routine returns. For routines that normally return right away, only one line is displayed, beginning with an asterisk (*).

Note

This facility does not trace QIOs and other system services.

1.2.7. Checking Name Server Operation

After verifying that the underlying transport is working, check to see whether the remote host can be reached by its host name. If your name server resides on a remote system, make sure your resolver configuration specifies that system. To determine whether the resolver is pointing to the correct server, enter the following command:

```
TCPIP> SHOW NAME_SERVICE

BIND Resolver Parameters

 Local domain: lkg.dec.com

 System

  State:     Started, Enabled

  Transport: UDP
  Domain:    lkg.dec.com
  Retry:     4
  Timeout:   4
  Servers:    rufus.lkg.dec.com, peach.lkg.dec.com
  Path:       lkg.dec.com

 Process

  State:     Enabled

  Transport:
  Domain:
  Retry:
  Timeout:
  Servers:
  Path:
```

Make sure the remote servers are reachable (using `ping`) and that they are valid name servers.

If your name server resides on the local system, use the SHOW NAME_SERVICE command to make sure your resolver points to `localhost`.

Next, verify that the TCPIP$BIND process is enabled and running. First, enter the following command to determine whether TCPIP$BIND is enabled:

```
TCPIP> SHOW SERVICE

Service           Port  Proto    Process          Address         State

BIND                53  TCP,UDP  TCPIP$BIND       0.0.0.0         Enabled
DHCP                67  UDP      TCPIP$DHCP       0.0.0.0         Enabled
DIOSERVER         1451  TCP      CLM              0.0.0.0         Disabled
ECHO                 7  TCP      MULTI            0.0.0.0         Disabled
ESNMP              705  UDP      ESNMP            0.0.0.0         Disabled
FINGER              79  TCP      TCPIP$FINGER     0.0.0.0         Enabled
FTP                 21  TCP      TCPIP$FTP        0.0.0.0         Enabled
HELLO            12345  TCP      HELLO_WORLD      0.0.0.0         Disabled
JOHN               520  UDP      UCX$ROUTER       0.0.0.0         Disabled
LBROKER           6570  UDP      TCPIP$LBROKER    0.0.0.0         Disabled
```

```
LPD                  515  TCP       TCPIP$LPD         0.0.0.0          Enabled
MATT                5432  TCP       TCPIP$RLOGIN      0.0.0.0          Disabled
METRIC               570  UDP       TCPIP$METRIC      0.0.0.0          Enabled
MOUNT                 10  TCP,UDP   TCPIP$MOUNTD      0.0.0.0          Enabled
NFS                 2049  UDP       TCPIP$NFS         0.0.0.0          Enabled
NOTES               3333  TCP       NOTESRVR          0.0.0.0          Enabled
NTP                  123  UDP       TCPIP$NTP         0.0.0.0          Enabled
PCNFS               5151  TCP,UDP   TCPIP$PCNFSD      0.0.0.0          Enabled
POP                  110  TCP       TCPIP$POP         0.0.0.0          Enabled
PORTMAPPER           111  TCP,UDP   TCPIP$PORTM       0.0.0.0          Enabled
REXEC                512  TCP       TCPIP$REXEC       0.0.0.0          Enabled
RLOGIN               513  TCP       not defined       0.0.0.0          Enabled
RSH                  514  TCP       TCPIP$RSH         0.0.0.0          Enabled
SMTP                  25  TCP       TCPIP$SMTP        0.0.0.0          Enabled
SNMP                 161  UDP       TCPIP$SNMP        0.0.0.0          Enabled
TELNET                23  TCP       not defined       0.0.0.0          Enabled
TFTP                  69  UDP       TCPIP$TFTP        0.0.0.0          Enabled
XDM                  177  UDP       TCPIP$XDM         0.0.0.0          Enabled
```

If the BIND process is enabled, it will appear in the display.

Then determine whether the BIND process is running by entering the following command:

```
$ SHOW SYSTEM /NETWORK
OpenVMS V7.1-1H2   on node RUFUS   27-JUN-2000 16:45:46.84  Uptime  16 01:55:35
  Pid    Process Name     State  Pri      I/O       CPU     Page flts  Pages
2FC0021F TCPIP$NTP        LEF     10  2042786   0 00:02:03.43     657    190  N
2FC00221 TCPIP$LBROKER    LEF      9  3779921   0 00:06:27.51     652    271  N
2FC05046 TCPIP$POP_1      HIB     10   243688   0 00:00:48.42     955    598  N
2FC00289 TCPIP$PORTM      LEF     10    13289   0 00:00:03.23     614    189  N
2FC0628F TCPIP$RE_BG1879  LEF      6     1647   0 00:00:00.96    1709    612  N
2FC0089A NFS$SERVER       LEF     10    89284   0 00:00:19.28     978    580  N
2FC06C9E NOTES$00CD_2*    HIB      6   208844   0 00:01:22.65    1932    152  N
2FC03EC7 TCPIP$BIND_1     LEF     10   515297   0 00:01:26.06     972    322  N
2FC01CF6 TCPIP$PCNFSD     LEF     10      326   0 00:00:00.27     660    228  N
```

If the TCPIP$BIND_1 process is not running, look for errors in the SYS$SPECIFIC:[TCPIP $BIND]TCPIP$BIND_RUN.LOG file.

To reduce the possibility of a name server being unavailable, you might configure more than one name server on your network. This way, if the primary name server is unreachable or unresponsive, the resolver can query the other name server.

1.2.8. Checking the Route to a Remote Host

If you receive “network unreachable” messages, you may be experiencing routing problem. You can easily detect whether the problem is with your local routing table by doing the following:

- Enter a `netstat -rn` or SHOW ROUTE command.

  Display the routing table, then compare the output to the routing table of a properly running system. Make sure there is a default route defined and that the IP address listed in the gateway column for the default route and the local host are in the same subnet. The default route specifies the gateway to use when a route is not explicitly defined for the destination IP address.

For example, enter the following command:

```
TCPIP> netstat -rn

Routing tables
Destination      Gateway            Flags     Refs     Use Interface

Route Tree for Protocol Family 2
default          16.20.0.173        UG         17  1526068  WE0
10.10/16         16.20.208.154      UGS         0   204911  WE0
10.10.39/25      10.10.39.2         U           2    17942  BE0
16.20/16         16.20.208.100      U          45  6219676  WE0
16.20/16         16.20.208.208      U           0        0  WE0
127.0.0.1        127.0.0.1          UH          1    69844  LO0

Route Tree for Protocol Family 26
::1              Link#1             UH          0        0  LO0
ff01::/16        Link#1             U           0        0  LO0
```

- To display a default route using the VSI TCP/IP Services management commands, enter one of the following commands:

```
$ TCPIP SHOW ROUTE /PERMANENT /DEFAULT
$ TCPIP SHOW ROUTE /DEFAULT
```

The following example shows typical output from these two commands:

```
$ TCPIP SHOW ROUTE /PERMANENT /DEFAULT

                              PERMANENT

Type           Destination                              Gateway

PN    0.0.0.0                                       rufus.lkg.dec.com


$ TCPIP SHOW ROUTE /DEFAULT

                              DYNAMIC

Type           Destination                              Gateway

DN    0.0.0.0                                  10.10.2.66
$
```

To set a default route, enter a command similar to the following:

```
$ TCPIP SET ROUTE /DEFAULT /GATE=n.n.n.n
```

You can also set a default route by running the TCPIP$CONFIG procedure and selecting option 1 for Core, and then option 3 for Routing. TCPIP$CONFIG prompts with:

```
* Do you want to configure dynamic ROUTED or GATED routing [NO]:
```

Take the default value by pressing the Enter key. TCPIP$CONFIG then displays the current configuration and asks whether you want to reconfigure a default route:

```
The current configuration for the default route is:

                                PERMANENT

Type           Destination                              Gateway

PN    0.0.0.0                                       rufus.lkg.dec.com

* Do you want to reconfigure a default route [YES]:
Enter the Default Gateway host name []:
```

- Next, use `ping` to see whether you can reach the routing gateway.

1.2.9. Checking the Routes Known to a Gateway

The `traceroute` command helps you locate problems between the local host and the remote destination by tracing the route of UDP packets from the local host to a remote host. Tracing attempts to determine the name and IP address of each gateway along the route to the remote host.

The `traceroute` command works by sending UDP packets with small time-to-live (TTL) values and an invalid port number to the remote system. The TTL values increase in increments of one for each group of three UDP packets sent. When a gateway receives a packet, it decrements the TTL. If the TTL is zero, the packet is not forwarded, and an ICMP “time exceeded” message is returned.

Intermediate gateways are detected when they return an ICMP “time exceeded” message. When `traceroute` receives an “invalid port” message, it knows that it reached the remote destination. (`traceroute` operates by intentionally using an invalid port.) When `traceroute` receives this message, it knows it has reached the destination host and terminates the trace. In this way, `traceroute` develops a list of gateways starting at one hop away, and increasing one hop at a time until the remote host is reached.

For more information about using `traceroute`, see Appendix A.

1.2.10. Determine Whether Network Services Are Available

The auxiliary server functions like the UNIX internet daemon (`inetd`) by managing access to the network services. The auxiliary server assigns standard port numbers to services such as the BOOTP, SMTP, or FTP servers, and starts the appropriate image after receiving an incoming request.

To verify correct operation of a service, you need to verify that the service:

- Has an entry in the service database
- Has the correct attributes defined
- Account has the correct privileges
- Is enabled
- Is started

1.2.10.1. Displaying the Service Database

To display the services database, enter the SHOW SERVICE command. For example:

```
TCPIP> SHOW SERVICE
  ❶                  ❷     ❸         ❹                ❺             ❻
Service             Port  Proto     Process          Address       State
FINGER                79  TCP       TCPIP$FINGER     0.0.0.0       Disabled
FTP                   21  TCP       TCPIP$FTP        0.0.0.0       Enabled
LPD                  515  TCP       TCPIP$LPD        0.0.0.0       Enabled
MOUNT                 10  UDP       TCPIP$NFS_M      0.0.0.0       Enabled
NFS                 2049  UDP       TCPIP$NFS        0.0.0.0       Enabled
NTP                  123  UDP       TCPIP$NTP        0.0.0.0       Enabled
PCNFS               5151  TCP,UDP   TCPIP$PCNFSD     0.0.0.0       Enabled
POP                  110  TCP       TCPIP$POP        0.0.0.0       Enabled
PORTMAPPER           111  TCP,UDP   TCPIP$PORTM      0.0.0.0       Enabled
REXEC                512  TCP       TCPIP$REXEC      0.0.0.0       Enabled
RLOGIN               513  TCP       not defined      0.0.0.0       Enabled
RSH                  514  TCP       TCPIP$RSH        0.0.0.0       Enabled
SMTP                  25  TCP       TCPIP$SMTP       0.0.0.0       Enabled
SNMP                 161  UDP       TCPIP$SNMP       0.0.0.0       Enabled
TELNET                23  TCP       not defined      0.0.0.0       Enabled
TFTP                  69  UDP       TCPIP$TFTP       0.0.0.0       Enabled
```

❶ This column lists those services with entries in the TCPIP services database. If not listed in this column, the service was never enabled during the configuration procedure (using TCPIP $CONFIG.COM). To enable additional services, run the TCPIP$CONFIG procedure.

❷ This column lists the port on which the service listens for connection requests. The port number is either the well-known port number for the service or an ephemeral port number assigned when the socket is assigned a protocol address.

❸ This column lists the TCP/IP protocol that the service uses to communicate with the client process.

❹ This column lists the process name for the service. If you use the DCL command SHOW SYSTEM /NETWORK, this is the process name you should see if the process is running.

❺ This column lists the IP address of the interface on which the service accepts connection requests. IP address 0.0.0.0 indicates that the service will accept connection requests received on any of the local interfaces.

❻ This column lists whether the service is enabled or disabled. The term enabled indicates that the next time VSI TCP/IP Services starts, VSI TCP/IP Services starts all services that are marked in the service database as enabled. In this example, of the services listed, all services except `finger` will start the next time VSI TCP/IP Services restarts.

Note

In this example, the `finger` service was configured with TCPIP$CONFIG. However, at some point, `finger` was disabled either by a TCPIP management command or by an incremental shutdown of the service.

1.2.10.2. Displaying Service Attributes

Each service should have the following items defined in the services database:

- OpenVMS user account (also in user authorization file [UAF])
- Unique port number
- Protocol
- Name and location of the startup command procedure and log file

- Service parameters (for example, timeouts, privileges)
- Flags

If these items are not defined correctly, or if the service account privileges and file protections are not assigned correctly, the service will fail to respond to an incoming request. This failure may be logged in the service-specific log file.

To display information about a service, enter the TCPIP command SHOW SERVICE /FULL and specify the service name. For example:

```
$ TCPIP TCPIP>
SHOW SERVICE /FULL TELNET

Service: TELNET                                                    ❶
                        State:     Enabled
Port:           23      Protocol:  TCP             Address:  0.0.0.0
Inactivity:      1      User_name: not defined     Process:  not defined
Limit:          57      Active:     12             Peak:      14

File:          not defined
Flags:         Listen Rtty

Socket Opts:  Keepalive Rcheck Scheck                              ❷
 Receive:         3000     Send:            3000

Log Opts:      Actv Dactv Conn Error Logi Logo Mdfy Rjct           ❸
 File:         not defined

Security                                                           ❹
 Reject msg:  not defined
 Accept host: 0.0.0.0
 Accept netw: 0.0.0.0
TCPIP>
```

❶ This section displays information about the service: service name, process name, user name, port and interface on which the service is listening, whether the service is enabled or disabled, and the number of copies of the service that can run at one time.

❷ This section displays the socket options that the service uses. The service's socket options can be changed dynamically, though it is unlikely that someone would change them. If you suspect that improper socket options are in effect, you can reestablish the default values by disabling the service, running TCPIP$CONFIG, and then enabling the service.

❸ This section displays the name of the log file that receives event messages and the events that the service will log. Checking the log file may indicate the cause of a problem.

❹ This security section displays a list of hosts and networks that are specifically given or denied access to the service. If one system is unable to access a service, check this section to see whether the system or its associated network is being denied the service.

1.2.10.3. Verifying Process Privileges

To check the privileges associated with a service's process, enter a command for the process, as follows:

```
$ INSTALL LIST/FULL TCPIP$SMTP_RECEIVER

DISK$VMS721:<SYS0.SYSCOMMON.SYSEXE>.EXE
   TCPIP$SMTP_RECEIVER;1
                  Open Hdr Shared   Prv
        Entry access count         = 20
        Current / Maximum shared   = 1 / 1
        Global section count       = 1
        Privileges = SYSPRV
        Authorized = SYSPRV

$ INSTALL LIST/FULL TCPIP$FTP_CHILD

DISK$VMS721:<SYS0.SYSCOMMON.SYSEXE>.EXE
   TCPIP$FTP_CHILD;1
                  Open Hdr Shared   Prv
        Entry access count         = 42
        Current / Maximum shared   = 1 / 3
        Global section count       = 1
        Privileges = PSWAPM OPER
        Authorized = PSWAPM OPER
```

1.2.10.4. Verifying Account Privileges

To determine the privileges associated with the service's account, run the OpenVMS Authorize utility and then use the SHOW command with the process name of the service, as follows:

```
A72KT: SET DEFAULT SYS$SYSTEM
A72KT: RUN AUTHORIZE
UAF> SHOW TCPIP$SNMP

Username: TCPIP$SNMP                        Owner:  TCPIP$SNMP
Account:  TCPIP                             UIC:    [3655,13] ([TCPIP$AUX,TCPIP$S
NMP])
CLI:      DCL                               Tables: DCLTABLES
Default:  SYS$SYSDEVICE:[TCPIP$SNMP]
LGICMD:   LOGIN
Flags:  Restricted
Primary days:   Mon Tue Wed Thu Fri
Secondary days:                   Sat Sun
Primary   000000000011111111112222  Secondary 000000000011111111112222
Day Hours 012345678901234567890123  Day Hours 012345678901234567890123
Network:  ##### Full access ######            ##### Full access ######
Batch:    -----  No access  ------            -----  No access  ------
Local:    -----  No access  ------            -----  No access  ------
Dialup:   -----  No access  ------            -----  No access  ------
Remote:   -----  No access  ------            -----  No access  ------
Expiration:            (none)    Pwdminimum:  6   Login Fails:     0
Pwdlifetime:         90 00:00    Pwdchange:      (pre-expired)
Last Login:            (none) (interactive),  7-AUG-2000 12:45 (non-interactive)
Maxjobs:          0  Fillm:        50  Bytlm:        52200
Maxacctjobs:      0  Shrfillm:      0  Pbytlm:           0
Maxdetach:        0  BIOlm:        18  JTquota:       4096
Prclm:            8  DIOlm:        18  WSdef:          350
Prio:             8  ASTlm:       100  WSquo:          512
Queprio:          4  TQElm:        15  WSextent:       512
CPU:         (none)  Enqlm:       100  Pgflquo:      10240
Authorized Privileges:
  NETMBX       TMPMBX
Default Privileges:
  NETMBX       TMPMBX
```

1.2.10.5. Looking for OPCOM Messages

The following is another method of detecting failure of the auxiliary server to start a service:

1. Enter the following commands:

```
$ SET PROCESS /PRIVILEGE=OPER          ❶
$ SET AUDIT /ALARM /ENABLE=FAILURE     ❷
$ REPLY/ENABLE=NETWORK                 ❸
```

   ❶ Enables your process with operator privileges.
   ❷ Enables security auditing to log unsuccessful object attempts.
   ❸ Establishes your terminal as an operator's terminal.

2. After you enter these commands, have a remote user try to connect to the service on the local system.

3. Review any OPCOM messages for indications of the problem.

1.3. Using Online Help

CP/IP Services provides online help for commands, utilities, and error messages.

For information about using TCP/IP management commands, enter the HELP command from the TCP/IP management control program.

For example, to display online help about using the sysconfig command, enter the following commands:

```
$ TCPIP
TCPIP> HELP SYSCONFIG
```

For information about TCP/IP user commands, enter the HELP command at the DCL prompt. For example, to display online help about the FTP command, enter the following command:

```
$ HELP FTP
```

Some user applications provide additional information from the user interface. For example:

```
$ FTP
FTP> HELP
```

Commands for obtaining online information from the application user interface are application specific. Refer to the description of the application in the *VSI TCP/IP Services for OpenVMS User's Guide* for specific instructions.

To display information about a TCP/IP error message, enter the HELP/MESSAGE command, followed by the error message name. For example, if you receive the FTP_CONHST error when you try to establish an FTP connection, enter the following command to get information about how to respond to the error:

```
$ HELP/MESSAGE FTP_CONHST

 FTP_CONHST,  error connecting to remote host 'host'

  Facility:     TCPIP, FTP Server

  Explanation:  An error occurred in the FTP client. A network error occurred
```

```
                or the network is not active.

  User Action:  Make sure network communication is active on the system.
```

1.4. Using OpenVMS ANALYZE Extensions

TCP/IP Services provides extensions to the OpenVMS DCL command ANALYZE, allowing you to analyze TCP/IP operations on the system. To obtain information about using the TCP/IP extensions to the SDA interface, enter the following commands:

```
$ ANALYZE/SYSTEM
SDA> TCPIP HELP
```

Chapter 2. Tuning Techniques

This chapter describes how to change system operation to solve specific networking problems. Once you have diagnosed the cause of the problem, as described in Chapter 1 [http://odl.sysworks.biz/disk$axpdocjun042/network/tcpip54/trouble/6631pro.html#ttips_chap], use the information in this chapter to solve the problem.

The tuning techniques described in this chapter include:

- Modifying subsystem attributes (Section 2.1)
- Tuning server applications (Section 2.2.2)
- Solving server performance problems (Section 2.3)

2.1. Subsystem Attributes

TCP/IP Services supports UNIX subsystems and allows you to modify the attributes of those subsystems to change the way the TCP/IP Services software operates.

Subsystem configuration is provided for compatibility with UNIX systems. VSI strongly advises you not to modify the attributes associated with subsystems except when the adjustment of an attribute is indicated (for example, to improve performance). In most cases, corresponding TCP/IP management commands are provided to help limit the side effects of modifying subsystem attributes.

The following sections describe how to display and modify the settings of the subsystem attributes. Modifying subsystem attributes without full knowledge of possible effects can cause unpredictable results and is recommended only as specifically directed by VSI.

2.1.1. Displaying Subsystems and Attributes

To display all the kernel subsystems and their status on the system, enter the following commands:

```
$ TCPIP
TCPIP> sysconfig -s
cm: loaded and configured
inet: loaded and configured
iptunnel: loaded and configured
ipv6: loaded and configured
net: loaded and configured
snmpinfo: loaded and configured
socket: loaded and configured
inetkvci: loaded and configured
proxy: loaded and configured
nfs: loaded and configured
vfs: loaded and configured
TCPIP>
```

Note

Depending on the configuration of your system, the list of subsystems displayed may differ from this example.

2.1.1.1. Static and Dynamic Subsystems

There are two types of subsystems:

- **Static** subsystems are loaded at startup time and can be unloaded only when TCP/IP shuts down.
- **Dynamic** subsystems can be loaded and unloaded at will without shutting down and restarting TCP/IP Services.

You can use the `sysconfig -m` command to display static and dynamic subsystems, as follows:

```
$ TCPIP
TCPIP> sysconfig -m
cm: static
inet: static
iptunnel: static
ipv6: static
net: static
snmpinfo: static
socket: static
inetkvci: static
proxy: static
nfs: static
vfs: static
```

Subsystems can be loaded but not available for use. Subsystems can have the following states:

- Loaded and configured (available for use)
- Loaded and unconfigured (not available for use)

  This state applies only to static subsystems, which you can unconfigure, but you cannot unload.
- Unloaded (not available for use)

  This state applies only to loadable subsystems, which are automatically unloaded when you unconfigure them.

2.1.1.2. Displaying the Attribute Values for a Subsystem

Each subsystem contains a set of attributes with associated values.

To display the attributes of a particular subsystem, include the `-q` option and the name of the subsystem. For example, to display the attributes of the `socket` subsystem, enter the following command:

```
TCPIP> sysconfig -q socket
socket:
sbcompress_threshold = 0
sb_max = 1048576
sobacklog_drops = 0
sobacklog_hiwat = 1
soinp_reuse = 1
somaxconn = 1024
somaxconn_drops = 0
sominconn = 1
TCPIP>
```

Subsystem attributes are specific to OpenVMS. Some UNIX attributes area not supported on OpenVMS systems.

To determine whether an attribute is supported, use the -q option and specify both the subsystem and the attribute.

If the subsystem is not configured, sysconfig displays a message similar to the following:

```
TCPIP> sysconfig -q inet tcbhashsize
framework error: subsystem 'inet' not found
```

To display the maximum and minimum values for attributes, include the -Q option and specify both the subsystem and the attribute. If you do not specify an attribute, the system displays all the attributes for the specified subsystem.

Note

On OpenVMS, you must enclose any uppercase options in quotation marks.

For example, to display the minimum and maximum values for the `rexmtmax` attribute in the inet subsystem, enter the following command:

```
TCPIP> sysconfig "-Q" inet tcp_rexmtmax
inet:
tcp_rexmtmax -  type=INT op=CRQ min_val=1 max_val=2147483647
TCPIP>
```

The `sysconfig "-Q"` command also displays the op code, which can consist of one or more of the following:

- C, which indicates that the attribute can be configured
- R, which indicates that the attribute can be reconfigured
- Q, which indicates that the attribute can be queried

2.1.2. Modifying Subsystem Attribute Values

You can modify the values of subsystem attributes either temporarily or permanently.

If you change the value of a subsystem attribute at run time, the change will persist only as long as the system continues to run. When you reboot the system, the attribute value reverts to the previous setting.

For temporary modifications that will not persist across reboots, use the `sysconfig -r` command as described in Section 2.1.2.1.

To modify an attribute value permanently, include the attribute in the system configuration table, sysconfigtab. Values specified in sysconfigtab are preserved through system reboots. To include an attribute in the sysconfigtab file, use the sysconfigdb utility, as described in Section 2.1.3.

Note

Always use the recommended utility to modify subsystem attributes. Do not modify the `sysconfigtab` file directly.

2.1.2.1. Reconfiguring Attributes

You can reconfigure an attribute (that is, modify its value temporarily), using the `sysconfig` command, if the attribute is reconfigurable (the `op` code includes R). This allows you to determine whether modifying an attribute will improve your system performance. Temporary modifications are lost when you reboot the system.

To temporarily modify an attribute's current value, use the `sysconfig` command with the `-r` option. Specify both the subsystem and the attribute on the command line:

For example, to modify the `tcp_keepinit` value to be 30 temporarily, enter the following command:

```
TCPIP> sysconfig -r inet tcp_keepinit=30
tcp_keepinit: reconfigured
TCPIP>
```

After modifying the attribute value, you must either reload the subsystem or restart TCP/IP Services, depending on whether the attribute is part of a dynamic subsystem or a static subsystem.

- Unload the dynamic subsystem using the `sysconfig -u` command. Then use the `sysconfig -r` command to reload the subsystem.
- Reload static subsystems by stopping and restarting TCP/IP Services.

2.1.3. Configuring Attributes

You can configure an attribute (that is, modify subsystem attributes so that changes persist across reboots), by storing the attribute and its value in the system configuration table (TCPIP $ETC:SYSCONFIGTAB.DAT), if the attribute's op code includes C.

When a subsystem is loaded, the attributes that are not listed in the SYSCONFIGTAB.DAT file are set to their default values.

To modify subsystem attributes in the SYSCONFIGTAB.DAT file, follow these steps:

1. Use the `sysconfigdb` utility to update the system configuration table, as described in Section 2.1.3.1
2. Reload the subsystem. A dynamic subsystem can be unloaded and reloaded using the `sysconfig` utility. A static subsystem is reloaded when the TCP/IP Services software is restarted.

2.1.3.1. Format of the SYSCONFIGTAB File

The TCPIP$ETC:SYSCONFIGTAB.DAT file is an ASCII text file in stanza file format. Do not edit the `sysconfigtab` file directly. Always use the recommended commands to modify attribute values in the `sysconfigtab` file.

An entry in the `sysconfigtab` file is formatted as follows:

```
subsystem1:
        parameter1=value1 parameter1=value2
subsystem2:
        parameter1=value1 parameter1=value2
```

To modify a subsystem attribute permanently, create a stanza-formatted file in your own directory. In the following example, the stanza file is named SOCKET_ATTRS.TXT.

```
$ TYPE SOCKET_ATTRS.TXT

socket:
         socket_param1 = socket_value1
$
```

After you create the stanza file, update the system configuration table using the `sysconfigdb` utility.

To run the `sysconfigdb` utility, enter the following commands:

```
$ TCPIP
TCPIP> sysconfigdb
```

To update the system configuration table, use the `sysconfigdb` command with the `-u` option. Specify the stanza file on the command line using the `-f` option. The following is the format of the `sysconfigdb` command:

```
TCPIP> sysconfigdb -u -f stanza-filename subsystem
```

In this command line, *stanza-filename* is the file name of the stanza file that you created. The value for *subsystem* is the subsystem name for which you are changing an attribute.

The `sysconfigdb` command reads the specified file and updates the `sysconfigtab` file. The modifications are made to the subsystem when it is reloaded.

For example, the following stanza file (TABLE_MGR.STANZA) defines the attributes for two subsystems, TABLE_MGR_1 and TBL_MGR_2.

```
$ TYPE TABLE_MGR.STANZA

table_mgr_1:
            size = 10
            name = Ten-Element-Table
tbl_mgr_2:
            size = 5
            name = Five-Element-Table

$
```

To add the contents of this stanza file to the system configuration table, enter the following commands:

```
$ TCPIP
TCPIP> sysconfigdb -a -f table_mgr.stanza table_mgr_1

TCPIP> sysconfigdb -a -f table_mgr.stanza tbl_mgr_2
```

This example does not change the value of attributes on the running system. For information about changing attribute values on the running system, see Section 2.1.2.1.

For complete information about using the `sysconfigdb` command, refer to Appendix A.

2.1.3.2. Stanza File Format

The syntax for a stanza file entry is as follows:

```
entry-name:
       Attribute1-name = Attribute1-value
       Attribute2-name = Attribute2-value
       Attribute3-name = Attribute3-value1, Attribute3-value2
       .
       .
       .
```

The *entry-name* variable specifies the subsystem name.

The attributes for the subsystem are specified with the *Attribute1-name*, *Attribute2-name*, and *Attribute3-name* variables.

The values for the attributes are specified with the *Attribute1-value*, *Attribute2-value*, *Attribute3-value1*, and *Attribute3-value2* variables.

The stanza file syntax rules are as follows:

- Separate entries by one or more blank lines.
- A colon (:) terminates an entry name.
- A new line terminates an attribute name and value pair.
- Separate a attribute name and attribute value with an equals sign (=).
- Separate more than one attribute value with a comma (,).
- Entry names and attribute names can contain any printable character except spaces, new lines, and special characters, which must be specified appropriately.
- Entry attribute values can contain any printable character except new lines and special characters, which must be specified appropriately.
- Spaces and tabs are allowed at the beginning and at the end of lines.
- A pound sign (#) at the beginning of a line indicates a comment.
- Comments should be included only at the beginning or the end of an entry.

Several special quoting characters allow attribute values to contain special values and data representations. If you specify a quoting character, surround the attribute value with quotation marks. For example, to specify an octal value, use the backslash character:

```
\007
```

2.1.4. Modifying Kernel Subsystems

Most resources used by the network subsystem are allocated and adjusted dynamically. However, you can make some adjustments to improve performance.

Table 2.1 summarizes the adjustments you can make, lists performance benefits and the adjustments that will achieve them, along with the tradeoffs (where applicable) associated with each adjustment.

Table 2.1. Network Tuning Guidelines

Performance Benefit	Tuning Adjustment	Tradeoff
Reduce the number of dropped incoming connection requests.	Increase the maximum number of pending TCP connections (Section 2.1.5.1).	Consumes memory resources.
Allow each server socket to handle more SYN packets simultaneously.	Increase the minimum number of pending TCP connections (Section 2.1.5.2).	Consumes memory resources.
Allow for a larger socket buffer.	Increase the maximum socket buffer size (Section 2.1.5.3).	Consumes memory. If you have a large number of sockets, memory consumption could be of concern.
Improve the TCP control block lookup rate and increase the raw connection rate.	Increase the size of the hash table that the kernel uses to look up TCP control blocks (Section 2.1.6.1).	Slightly increases the amount of pooled memory.
Reduce hash table lock contention for SMP systems.	Increase the number of TCP hash tables (Section 2.1.6.2).	Slightly increases the amount of pooled memory.
Improve performance on systems that use large numbers of interface alias.	Increase the size of the kernel interface alias table (Section 2.1.6.3).	None.
Allow partial connections to time out sooner, preventing the socket listen queue from filling up with SYN packets.	Increase the TCP partial connection timeout rate (Section 2.1.6.4).	Setting the `tcp_keepinit` value too low can cause connections to be broken prematurely.
Prevent premature retransmissions and decrease congestion.	Reduce the TCP retransmission rate (Section 2.1.6.5).	A long retransmit time is not appropriate for all configurations.
Clean up sockets that do not exit cleanly when the `keepalive` interval expires.	Enable TCP `keepalive` functionality (Section 2.1.6.6).	None.
Free connection resources sooner.	Make the TCP connection context time out more quickly at the end of the connection (Section 2.1.6.7).	Reducing the timeout limit increases the potential for data corruption; use caution if you make this adjustment.
Provide TCP and UDP applications with a specific range of ports.	Modify the range of outgoing connection ports (Section 2.1.6.9).	None.
Improve the efficiency of servers that handle remote traffic from many clients.	Disable the use of a PMTU (Section 2.1.6.10).	May reduce server efficiency for LAN traffic.

Performance Benefit	Tuning Adjustment	Tradeoff
Allow large socket buffer sizes.	Increase the maximum size of a socket buffer (Section 2.1.5.3).	Consumes memory resources.
Improve handling of a large number of BG devices.	Enable fast device creation and deletion (Section 2.2.4).	Consumes memory resources.

The following sections describe in detail how to modify `socket` and `inet` subsystem attributes.

2.1.5. Modifying Socket Subsystem Attributes

The `socket` subsystem attributes control the maximum number of pending connection attempts per server socket (that is, the maximum depth of the listen or SYN queue) and other behavior. You may be able to improve server performance by modifying the `socket` subsystem attributes described in Table 2.2.

Table 2.2. `socket` Subsystem Attributes

Attribute	Description
`somaxconn`	Controls the maximum number of pending TCP connections.
`sominconn`	Controls the minimum number of pending TCP connections.
`sb_max`	Controls the maximum size of a socket buffer.

In addition, the `socket` subsystem attributes `sobacklog_hiwat`, `sobacklog_drops`, and `somaxconn_drops` track events related to socket listen queues. By monitoring these attributes, you can determine whether the queues are overflowing.

2.1.5.1. Increasing the Maximum Number of Pending TCP Connections

The `socket` subsystem attribute `somaxconn` specifies the maximum number of pending TCP connections (the socket listen queue limit) for each server socket (for example, for the HTTP server socket). Busy servers often experience large numbers of pending connections. If the listen queue connection limit is too small, incoming connection requests may be dropped. Pending TCP connections can be caused by lost packets in the internet or denial of service attacks.

The default value for `somaxconn` is 1024.

VSI recommends increasing the `somaxconn` attribute to the maximum value, except on low-memory systems. The maximum value is 65535. Specifying a value that is higher than the maximum value can cause unpredictable behavior.

2.1.5.2. Increasing the Minimum Number of Pending TCP Connections

The `socket` subsystem attribute `sominconn` specifies the minimum number of pending TCP connections (backlog) for each server socket. This attribute controls how many SYN packets can be handled simultaneously before additional requests are discarded. Network performance can degrade if a client saturates a socket listen queue with erroneous TCP SYN packets, effectively blocking other users from the queue.

The value of the sominconn attribute overrides the application-specific backlog value, which may be set too low for some server software. If you do not have your application source code, you can use the sominconn attribute to set a sufficient pending-connection quota.

The default value is 0.

VSI recommends increasing the value of the sominconn attribute to the maximum value of 65535. The value of the sominconn attribute should be the same as the value of the somaxconn attribute (see Section 2.1.5.1).

2.1.5.3. Increasing the Maximum Size of a Socket Buffer

The socket subsystem attribute sb_max specifies the maximum size of a socket buffer.

Performance Benefits and Tradeoffs

Increasing the maximum size of a socket buffer may improve performance if your applications can benefit from a large buffer size.

You can modify the sb_max attribute without rebooting the system.

When to Tune

If you require a large socket buffer, increase the maximum socket buffer size.

Recommended Values

The default value of the sb_max attribute is 128 KB. Increase this value before you increase the size of the transmit and receive socket buffers.

2.1.6. Modifying Internet Subsystem Attributes

You may be able to improve inet subsystem performance by modifying the attributes described in Table 2.3.

Table 2.3. inet Subsystem Attributes

Attribute	Description
tcbhashsize	Controls the size of a TCP hash table.
tcbhashnum	Specifies the number of TCP hash tables.
inifaddr_hsize	Controls the size of the kernel interface alias table.
tcp_keepinit	Specifies the TCP partial connection timeout rate.
tcp_rexmit_interval_min	Specifies the rate of TCP retransmissions.
tcp_keepalive_default	Enables or disables the TCP keepalive function.
tcp_msl	Specifies the TCP connection context timeout rate.
tcp_nodelack	Delays acknowledgment messages after the receipt of network frames.
ipport_userreserved	Specifies the maximum value for the range of outgoing connection ports.
ipport_userreserved_min	Specifies the minimum value for the range of outgoing connection ports.

Attribute	Description
pmtu_enabled	Enables or disables use of the PMTU protocol.
ipqmaxlen	Prevents dropped input packets.

2.1.6.1. Increasing the Size of a TCP Hash Table

You can modify the size of the hash table that the kernel uses to look up Transmission Control Protocol (TCP) control blocks. The inet subsystem attribute tcbhashsize specifies the number of hash buckets in the kernel TCP connection table (the number of buckets in the inpcb hashtable).

Performance Benefits and Tradeoffs

The kernel must look up the connection block for every TCP packet it receives, so increasing the size of the table can speed the search and improve performance. This results in a small increase in pooled memory.

You can modify the tcbhashsize attribute without rebooting the system.

When to Tune

Increase the number of hash buckets in the kernel TCP connection table if you have an server.

Recommended Values

The default value of the tcbhashsize attribute is 512. For Internet servers, set the tcbhashsize attribute to 16384.

2.1.6.2. Increasing the Number of TCP Hash Tables

You can increase the number of hash tables the kernel uses to look up TCP control blocks. Because the kernel must look up the connection block for every Transmission Control Protocol (TCP) packet it receives, a bottleneck may occur at the TCP hash table in SMP systems. Increasing the number of tables distributes the load and may improve performance. The inet subsystem attribute tcbhashnum specifies the number of TCP hash tables.

Performance Benefits and Tradeoffs

For SMP systems, you may be able to reduce hash table lock contention by increasing the number of hash tables that the kernel uses to look up TCP control blocks. This will slightly increase pooled memory.

You cannot modify the tcbhashnum attribute without rebooting the system.

When to Tune

Increase the number of TCP hash tables if you have an SMP system that is an server.

Recommended Values

The minimum value of the tcbhashnum attribute is 1 (the default). The maximum value is 64. For busy server SMP systems, you can increase the value of the tcbhashnum attribute to 16. If you increase this attribute, you should also increase the size of the hash table by a similar factor. See Section 2.1.6.1 for more information.

2.1.6.3. Increasing the Size of the Kernel Interface Alias Table

The `inet` subsystem attribute `inifaddr_hsize` specifies the number of hash buckets in the kernel interface alias table (`in_ifaddr`).

If a system is used as a server for many different server domain names, each of which is bound to a unique IP address, the code that matches arriving packets to the right server address uses the hash table to speed lookup operations for the IP addresses.

Performance Benefits and Tradeoffs

Increasing the number of hash buckets in the table can improve performance on systems that use large numbers of aliases.

When to Tune

Increase the number of hash buckets in the kernel interface alias table if your system uses large numbers of aliases.

You can modify the `inifaddr_hsize` attribute without rebooting the system.

Recommended Values

The default value of the `inet` subsystem attribute `inifaddr_hsize` is 32; the maximum value is 512.

For the best performance, the value of the `inifaddr_hsize` attribute is always rounded down to the nearest power of 2. If you are using more than 500 interface IP aliases, specify the maximum value of 512. If you are using fewer than 250 aliases, use the default value of 32. For a number of aliases between 250 and 500, use a value that is a power of 2 between 32 and 512.

2.1.6.4. Increasing the TCP Partial Connection Timeout Rate

If increasing the `somaxconn` limit does not prevent the listen queue from filling, or if the default grows to an excessive length, you can make partial connections time out sooner by decreasing the value of the `inet` subsystem attribute `tcp_keepinit`.

The `tcp_keepinit` attribute specifies the amount of time that a partial connection remains on the socket listen queue before it times out.

Performance Benefits and Tradeoffs

Network performance can degrade if a client overfills a socket listen queue with TCP SYN packets, thereby blocking other users from the queue. To eliminate this problem, increase the value of the `sominconn` attribute to its maximum value. If the system continues to drop SYN packets, decrease the value of the `tcp_keepinit` attribute to 30 (15 seconds). Monitor the values of the `sobacklog_drops` and `somaxconn_drops` attributes to determine whether the system is dropping packets. (See Section 2.1.7 for more information about event counters.)

You can modify the `tcp_keepinit` attribute without rebooting the system.

When to Tune

Modify the TCP partial-connection timeout limit only if the value of the `somaxconn_drops` attribute increases often. If this occurs, decrease the value of the `tcp_keepinit` attribute.

Recommended Values

The value of the `tcp_keepinit` attribute is in units of 0.5 seconds. The default value is 150 units (75 seconds). If the value of the `sominconn` attribute is 65535, use the default value of the `tcp_keepinit` attribute.

If you set the value of the `tcp_keepinit` attribute too low, you may prematurely break connections associated with clients on network paths that are slow or network paths that lose many packets. Do not set the value to less than 20 units (10 seconds).

2.1.6.5. Slowing TCP Retransmission Rate

The `inet` subsystem attribute `tcp_rexmit_interval_min` specifies the minimum amount of time before the first TCP retransmission.

Performance Benefits and Tradeoffs

You can increase the value of the `tcp_rexmit_interval_min` attribute to slow the rate of TCP retransmissions, which decreases congestion and improves performance.

You can modify the `tcp_rexmit_interval_min` attribute without rebooting the system.

When to Tune

Not every connection needs a long retransmission time. Usually, the default value is adequate. However, for some wide area networks (WANs), the default retransmission interval may be too small, causing premature retransmission timeouts. This may lead to duplicate transmission of packets and the erroneous invocation of the TCP congestion-control algorithms.

To check for retransmissions, use the `netstat -p tcp` command and examine the output for data packets retransmitted.

Recommended Values

The `tcp_rexmit_interval_min` attribute is specified in units of 0.5 second. The default value is 2 units (1 second).

Do not specify a value that is less than 1 unit. Do not change the attribute unless you fully understand TCP algorithms and your network topology.

2.1.6.6. Enabling the TCP Keepalive Function

The keepalive function enables the periodic transmission of messages on a connected socket in order to keep connections active. Sockets that do not exit cleanly are cleaned up when the keepalive interval expires. If keepalive is not enabled, those sockets continue to exist until you reboot the system.

Applications enable keepalive for sockets by setting the `setsockopt` function's

`SO_KEEPALIVE`

option. To override programs that do not set `keepalive`, or if you do not have access to the application sources, use the `inet` subsystem attribute `tcp_keepalive_default` to enable keepalive functionality.

Performance Benefit

Keepalive functionality cleans up sockets that do not exit cleanly when the keepalive interval expires.

You can modify the `tcp_keepalive_default` attribute without rebooting the system. However, sockets that already exist will continue to use old behavior, until the applications are restarted.

When to Tune

Enable keepalive if you require this functionality, and you do not have access to the source code.

Recommended Values

To override programs that do not set keepalive, or if you do not have access to application source code, set the `inet` subsystem attribute `tcp_keepalive_default` to 1 in order to enable keepalive for all sockets.

If you enable keepalive, you can also configure the following TCP options listed in Table 2.4 for each socket:

Table 2.4. TCP Keepalive Options

Option	Description
`tcp_keepidle`	Specifies the amount of idle time, in seconds, before sending a keepalive probe. The default interval is two hours.
`tcp_keepintvl`	Specifies the amount of time, in seconds, between retransmission of keepalive probes. The default interval is 75 seconds.
`tcp_keepcnt`	Specifies the maximum number of keepalive probes that are sent before the connection is dropped. The default is 8 probes.
`tcp_keepinit`	Specifies the maximum amount of time, in seconds, before an initial connection attempt times out. The default is 75 seconds.

2.1.6.7. Increasing the Timeout Rate for TCP Connection Context

The TCP protocol includes a concept known as the Maximum Segment Lifetime (MSL). When a TCP connection enters the `TIME_WAIT` state, it must remain in this state for twice the value of the MSL; otherwise, undetected data errors on future connections can occur. The `inet` subsystem attribute `tcp_msl` determines the maximum lifetime of a TCP segment and the timeout value for the `TIME_WAIT` state.

In some situations, the default timeout value for the `TIME_WAIT` state (60 seconds) is too large, thereby reducing the value of the `tcp_msl` attribute frees connection resources sooner than the default setting.

Performance Benefits and Tradeoffs

You can decrease the value of the `tcp_msl` attribute to make the TCP connection context time out more quickly at the end of a connection. However, this will increase the chance of data corruption.

You can modify the `tcp_msl` attribute without rebooting the system.

When to Tune

Usually, you do not have to modify the timeout limit for the TCP connection context.

Recommended Values

The value of the `tcp_msl` attribute is set in units of 0.5 second. The default value is 60 units (30 seconds), which means that the TCP connection remains in TIME_WAIT state for 60 seconds, or twice the value of the MSL.

Do not reduce the value of the `tcp_msl` attribute unless you fully understand the design and behavior of your network and the TCP protocol. It is strongly recommended that you use the default value; otherwise, there is the potential for data corruption.

2.1.6.8. Disabling Delayed Acknowledgment

The TCP/IP software can send an acknowledgment packet for every frame received over the network. However, this is an inefficient mode of operation. The tcp_nodelack attribute controls the delay of acknowledgment messages.

When this attribute is set to 0 (the default), network traffic is greatly reduced. If you set this attribute to 1, an acknowledgment message is sent for every frame received, increasing network traffic and impacting the performance of the network server.

2.1.6.9. Modifying the Range of Outgoing Connection Ports

When a TCP or UDP application creates an outgoing connection, the kernel dynamically allocates a nonreserved port number for each connection. The kernel selects the port number from a range of values between the value of the `inet` subsystem attribute `ipport_userreserved_min` and the value of the `ipport_userreserved` attribute. Using the default values for these attributes, the range of outgoing ports starts at 49152 and stops at 65535.

Performance Benefits and Tradeoffs

Modifying the range of outgoing connections provides TCP and UDP applications with a specific range of ports.

You can modify the `ipport_userreserved_min` and `ipport_userreserved` attributes without rebooting the system.

When to Tune

If your system requires outgoing ports from a particular range, you can modify the values of the `ipport_userreserved_min` and `ipport_userreserved` attributes.

Recommended Values

The default value of the `ipport_userreserved_min` attribute is 49152. The default value of the `ipport_userreserved` is 65535. The maximum value of each attribute is 65535.

Do not reduce the `ipport_userreserved` attribute to a value that is less than 65535, and do not reduce the `ipport_userreserved_min` attribute to a value that is less than 49152.

2.1.6.10. Disabling Use of the PMTU Protocol

Packets transmitted between servers are fragmented into units of a specific size in order to ease transmission of the data over routers and small-packet networks, such as Ethernet networks. When the `inet` subsystem attribute `pmtu_enabled` is enabled (set to 1, which is the default behavior), the system determines the largest common path maximum transmission unit (PMTU) value between servers and uses it as the unit size. The system also creates a routing table entry for each client network that attempts to connect to the server.

Performance Benefits and Tradeoffs

If a server handles traffic among many remote clients, disabling the use of a PMTU can decrease the size of the kernel routing table, which improves server efficiency. However, on a server that handles local traffic and some remote traffic, disabling the use of a PMTU can degrade bandwidth.

When to Tune

If an server has poor performance and the routing table increases to more than 1000 entries, you should disable the use of PMTU. This is also recommended if you have a server that handles traffic among many remote clients.

Recommended Values

To disable the use of PMTU protocol, set the value of the `pmtu_enabled` attribute to 0.

2.1.7. Displaying Socket Statistics

The `socket` subsystem has three attributes that monitor socket listen queue events:

- The `sobacklog_hiwat` attribute counts the maximum number of pending requests to any server socket.
- The `sobacklog_drops` attribute counts the number of times the system dropped a received SYN packet because the number of queued SYN_RCVD connections for a socket equaled the socket's backlog limit.
- The `somaxconn_drops` attribute counts the number of times the system dropped a received SYN packet because the number of queued SYN_RCVD connections for the socket equaled the upper limit on the backlog length (`somaxconn` attribute).

The initial value of these attributes is 0. Use the

```
sysconfig -q socket
```

command to display the current attribute values. If the values show that the queues are overflowing, you may need to increase the socket listen queue limit.

The value of the `sominconn` attribute should equal the value of the `somaxconn` attribute. When these two attributes are equal, the value of `somaxconn_drops` will have the same value as `sobacklog_drops`.

However, if the value of the `sominconn` attribute is 0 (the default), and if one or more server applications uses an inadequate value for the backlog argument to its listen system call, the value of `sobacklog_drops` may increase at a rate that is faster than the rate at which the `somaxconn_drops` counter increases. If this occurs, you may want to increase the value of the `sominconn` attribute. See Section 2.1.5.2 for more information.

2.2. Tuning Server Applications

In addition to tuning TCP/IP Services kernel attributes, performance improvements can be made to server applications by:

- Ensuring adequate memory configuration
- Logging IP addresses

2.2.1. Configuring Memory for High Performance

Each connection to an server requires enough memory resources for the following:

- Kernel socket structure
- Internet protocol control block (`inpcb`) structure
- TCP control block structure
- Any socket buffer space that is needed as packets arrive and are consumed

These memory resources total 1 KB for each connection endpoint (not including the socket buffer space), which means 10 MB of memory is required in order to accommodate 10,000 connections.

Your server must have enough memory to handle demanding peak loads. As a rule of thumb, if you configure ten times more memory than the server requires on a busy day, you will have sufficient memory to handle occasional spikes of activity.

There are no limitations on a server's ability to handle millions of TCP connections if memory resources are available to service the connections. If memory is insufficient, the server will reject new connection requests until memory is available. Use the `netstat -m` command to monitor the memory that is currently being used by the network subsystem. See Section 1.2.4 for information about displaying memory statistics.

2.2.2. Logging IP Addresses

If your server application logs client host names, the application software may force the system to perform a reverse DNS lookup to obtain the client's host name. Reverse DNS lookups are time-intensive and can cause performance problems on servers with many clients.

Many applications can be modified to log client IP addresses instead of client host names. Logging IP addresses instead of host names may significantly improve the efficiency of the server. Consult the documentation provided by the server software vendor to determine how to disable the logging of client host names.

For example, you can obtain information about modifying Apache HTTP Server software from the Apache HTTP Server documentation site.

2.2.3. Increasing the Auxiliary Server Connection Limit

The auxiliary server handles a limited number of service invocations in a one-minute period of time. The default is a maximum of 500 connection requests. If the number of requests exceeds this limit, the auxiliary server will not accept additional requests for that service.

If your server receives more than eight requests per second for a service that is spawned by the auxiliary server (for example, POP-3, FTP, and SMTP servers), increase the default connection request limit. You can check the service's log file to determine if a service has been shut down. For example, the file SYS $SYSDEVICE:[TCPIP$POP]TCPIP$POP_RUN.LOG will contain information about the POP service.

Because the auxiliary server does not spawn any known HTTP server, the connection request limit does not affect HTTP service.

2.2.4. Increasing the Maximum Number of BG Devices

You can configure TCP/IP Services to create more than 10,000 devices. This is useful if the system is a web server.

The net subsystem attributes the provide information about allow you to modify the way TCP/IP Services handles BG devices are described in Table 2.5.

Table 2.5. `net` Subsystem Attributes

Attribute	Description
`ovms_unit_count`	Informational attribute that shows the number of BG devices that exist in the system. You cannot modify this attribute.
`ovms_unit_creates`	Information attribute that shows a running total of all the BG devices created since TCP/IP Services was started. You can use this information to see how often devices are being used. You cannot modify this attribute.
`ovms_unit_fast_credel`	Specifies fast creation and deletion of BG devices, useful on a system that uses many BG devices. Modifying this attribute can improve performance by reducing: • The impact of the device list scan on the CPU cache. • The time that the I/O mutex is held for write access during the SYS$ASSIGN( ) and SYS $DASSGN( ) system services. The default setting for this attribute is 0, or OFF. This attribute can affect the amount of virtual memory used. To calculate the effect of this attribute on virtual memory usage, add one to the the number of BG devices (ovms_unit_maximum) and multiply by 4. For example, with the default setting of 9999, the virtual memory is 40,000 bytes. At its maximum value of 32767, the virtual memory is 131,072 bytes. You can modify this attribute.
`ovms_unit_limit`	This informational attribute shows the calculated maximum number of concurrent BG device units

	that can coexist at any time given the values of the ovms_unit_maximum and ovms_unit_minimum attributes. Note that BG0 and BG1 are always reserved. You cannot modify this attribute.
ovms_unit_minimum	Specifies the lowest numbered device unit. All BG devices after BG0 will begin with this unit number. You can modify this attribute.
ovms_unit_maximum	Specifies the highest numbered device unit. It is possible there will be BG device numbers above this, but they are not available to users. You can specify a value from 9999 to 32767. You can modify this attribute.
ovms_unit_seed	This informational attribute shows the next BG unit number that will be created. If ovms_unit_fast_credel is enabled, a duplicate BG device unit is sensed immediately and the unit seed incremented. If ovms_unit_fast_credel is not enabled, a device scan will occur as it normally does with cloned devices.

To modify ovms_unit_fast_credel, ovms_unit_minimum, and ovms_unit_maximum, define them in the SYSCONFIGTAB.DAT file, as described in Section 2.1.2.

2.3. Solving Performance Problems

This section contains information that you can use to identify and solve server performance problems.

The following tasks can help you to solve performance problems:

- Monitor the server.

  See Section 1.2.4 for information about monitoring the network, the virtual memory subsystem, and network socket statistics.

- Apply any patches recommended for your operating system.

  See the Recommended Patch Table for information about operating system patches that can improve performance.

- Apply the kernel attribute values that are recommended for your type of system.

  See Section 2.3.1 for a list of attributes that can be tuned to improve performance.

- Prevent "forbidden" and "*url* could not load" messages on Netscape enterprise server systems.

2.3.1. Tuning Recommendations for a Primary Server

This section provides recommendations for tuning a server for optimal performance. These recommendations are applicable to most configurations and include the attribute value and a reference to additional information.

The primary recommendations for servers, (including web servers, proxy servers, gateway systems, and firewall systems) are as follows:

- Tune the following `socket` subsystem attributes:

```
somaxconn = 65535
sominconn = 65535
```

- Tune the following `inet` subsystem attributes:

```
pmtu_enabled = 0
```

For proxy servers, gateway systems, and firewall systems, also apply these additional recommendations:

- Modify the following socket subsystem attribute as follows:

```
sbcompress_threshold = 600
```

- Modify the following `inet` subsystem attribute as follows:

```
ipport_userreserved = 65000
```

2.3.2. Improving Gigabit Ethernet Performance

You can improve the performance of TCP/IP Services over Gigabit Ethernet (GbE) by configuring the system to use jumbo frames.

Jumbo frames (frames larger than the 1518 octets specified by the IEEE 802.3 standard) improve performance because more data can be transmitted in each frame, hence reducing the frame rate and interrupt load on a system. Jumbo frames must be configured on both the Gigabit Ethernet switch as well as the Gigabit Ethernet controllers on the host systems. Configure jumbo frames before you start TCP/IP Services on the OpenVMS server.

There are two ways to configure jumbo frames for Gigabit Ethernet controllers on OpenVMS:

- Including the following LANCP command in SYSTARTUP_VMS.COM:

```
$ mcr lancp set dev ewa/jumbo ! enable jumbo frames for GbE.
```

- Setting the SYSGEN parameter LAN_FLAGS in MODPARAMS.DAT and running AUTOGEN. For example:

```
LAN_FLAGS=64   ! Set bit 6 to enable jumbo frames on GbE, a dynamic
 parameter
```

  Restart TCP/IP Services after you enable jumbo frames dynamically.

To verify that TCP/IP Services now sees a jumbo frame size IPMTU, use a command similar to:

```
$ ifconfig we0
WE0: flags=c43<UP,BROADCAST,RUNNING,MULTICAST,SIMPLEX>
    *inet 10.0.0.4 netmask ffff0000 broadcast 10.0.255.255 ipmtu 9000
```

You can verify jumbo frame configuration using the tcpdump command. For example, consider two systems connected with Gigabit Ethernet, GIGA1 and GIGA2:

On GIGA1, enter the following command:

```
$ tcpdump -c 2 -t port telnet
```

On GIGA2, enter the following command:

```
$ TELNET giga1
```

The `tcpdump` command displays the two packets from the TCP connection establishment and reveals the negotiated maximum segment size (`mss`). For example, on a correctly configured system the tcpdump trace from the TELNET command shows:

```
$ tcpdump -c 2 -t port telnet
tcpdump: listening on
Filtering in user process
GIGA2.49188 > GIGA1.23: S 165176320:165176320(0) win 3000
<mss 8960,nop,wscale 0>

GIGA1.23 > GIGA2.49188: S 1890602256:1890602256(0) ack 165176321 win 3000
<mss 8960,nop,wscale 0>
2 packets (out of 5 examined)
```

Note that the requested mss of 8960 in the SYN packet is also accepted by the server in the SYN ACK packket. On earlier versions of OpenVMS, the jumbo-frame mss may be less than 8960 bytes.

If one of the systems is not configured with jumbo frames, the tcpdump utility shows the following:

```
$ tcpdump -c 2 -t port telnet
tcpdump: listening on
Filtering in user process
GIGA2.49187 > GIGA1.23: S 142920192:142920192(0) win 3000
<mss 8960,nop,wscale 0>

GIGA1.23 > GIGA2.49187: S 2953104448:2953104448(0) ack 142920193 win 4380
<mss 1460,nop,wscale 0>
2 packets (out of 5 examined)
```

The TELNET client requests an mss of 8960 bytes in the SYN packet, but the server responds with an mss of 1460 bytes in the SYN ACK packet. Therefore, GIGA1 is not correctly configured for jumbo frames.

Appendix A. Troubleshooting Utilities Reference

This appendix provides more information about the troubleshooting utilities described in this manual. It also describes the utilities used for isolating and resolving problems with your network and network software.

To invoke a utility as a command at the system prompt, execute the SYS$STARTUP:TCPIP $DEFINE_COMMANDS.COM file. Execution of this file defines each utility as a foreign command.

arp

arp — Displays and controls Address Resolution Protocol (ARP) tables.

Syntax

`arp [-u] [-n] hostname`

`arp -a [-u] [-n] [-i] hostname`

`arp -d hostname`

`arp -g hostname`

`arp -s [-u] hostname hardware_addr [temp] [pub] [trail]`

`arp -f filename`

Description

The `arp` command displays or modifies the current ARP entry for the host specified by *hostname*. The *hostname* value can be specified by name or IP address, using dotted-decimal notation.

With no flags, the program displays the current ARP entry for `hostname`.

The ARP tables can be displayed by any user, but only privileged users can modify them.

Flags

-a

Displays all current ARP entries.

-d *hostname*

Deletes the entry for *hostname* if the user entering the command is a privileged user.

-f *filename*

Reads entries from *filename* and adds those entries to the ARP tables. Use of this flag requires system privileges. Entries in the file have the following format:

`hostname hardware_addr [temp] [pub] [trail]`

Fields in this format are:

Option	Description
hostname	Specifies the remote host identified by the entry.
hardware_addr	Specifies the hardware address of the remote host. The address is given as 6 hexadecimal bytes separated by colons.
`temp`	Specifies that this ARP table entry is temporary. When this argument is not used, the table entry is permanent.
`pub`	Indicates that the table entry will be published and that the current system will act as an ARP server, responding to requests for *hostname* even though the host address is not its own.
`trail`	Indicates that the trailer encapsulation can be sent to this host.

-g *hostname*

Sends a gratuitous ARP packet. The value for *hostname* can be a local host name, alias, or IP address.

-i *hostname*

Displays the interface with which the ARP entry is associated.

-n *hostname*

Displays numeric IP addresses and hardware addresses only. When this flag is not specified, arp displays hostnames, numeric IP addresses, and hardware addresses.

-s *hostname hardware_addr* [temp] [pub]

Creates a single ARP entry for *hostname*. Use of this flag requires privileges. Fields in the format are:

hostname	Specifies the remote host identified by the entry.
hardware_addr	Specifies the hardware address of the remote given as 6 hexadecimal bytes separated by colons.
`temp`	Specifies that this ARP table entry is temporary. When this argument is not used, the table entry is permanent.
`pub`	Indicates that the table entry will be published and the current system will act as an ARP server, responding to requests for `hostname` even though the host address is not its own.

-u

Displays the MAC address in noncanonical form, with address bytes reversed and separated by a colon character (:). By default, all addresses are displayed in canonical form with address bytes separated by the hyphen character (-).

When used with the `-s` flag, this indicates the *hardware_addr* is specified in noncanonical form.

Examples

The following examples show how to use the `arp` command.

1. ```
   TCPIP> arp -a
   a71kt.lkg.dec.com (10.10.2.1) at aa-00-04-00-71-f8 stale
   v71kt.lkg.dec.com (10.10.2.3) at aa-00-04-00-70-f8 stale
   v72kt.lkg.dec.com (10.10.2.4) at aa-00-04-00-6d-f8
   tlab9.lkg.dec.com (10.10.2.11) at aa-00-04-00-42-11
   timber.lkg.dec.com (10.10.2.14) at aa-00-04-00-c9-f8
   ```

   This example shows how to display the ARP address-mapping tables for the local host.

2. ```
   TCPIP> arp -a -i
   a71kt.lkg.dec.com (10.10.2.1) at aa-00-04-00-71-f8 stale (WE0)
   v71kt.lkg.dec.com (10.10.2.3) at aa-00-04-00-70-f8 (WE0)
   v72kt.lkg.dec.com (10.10.2.4) at aa-00-04-00-6d-f8 stale (WE0)
   tlab9.lkg.dec.com (10.10.2.11) at aa-00-04-00-42-11 (WE0)
   timber.lkg.dec.com (10.10.2.14) at aa-00-04-00-c9-f8 (WE0)
   ```

   This example shows how to display the ARP address-mapping tables for the local host and the interface.

3. ```
   TCPIP> arp -s laszlo 08:00:2b:0f:44:23 temp
   ```

   This example shows how to add a single entry for the remote host `laszlo` to the ARP mapping tables temporarily. The address is considered canonical even though the bytes are separated by colons. For input, the `arp` command does not use the colon (:) and hyphen (-) characters to indicate whether the address is canonical or noncanonical. You must have system privileges to execute this command.

4. ```
   TCPIP> arp -u -s laszlo 10:00:d4:f0:22:c4 temp
   ```

   This example shows how to add a single entry for the remote host `laszlo` to the `arp` mapping tables temporarily. The `-u` flag indicates the address is noncanonical. You must have system privileges to execute this command.

5. ```
   TCPIP> arp -f newentries
   ```

   This example shows how to add multiple entries to the ARP mapping tables from a file named `newentries`. You must have system privileges to execute this command.

dig

dig — Sends domain name query packets to name servers.

Syntax

```
dig [@server] domain[query-type] [query-class] [+query-option]
[-dig-option] [%comment]
```
```

Description

Domain Information Groper (`dig`) is a flexible command line utility you can use to gather information from Domain Name System servers. The `dig` utility has two modes: simple interactive mode, which makes a single query; and batch mode, which executes a query for each query in a list of several query lines. All query options are accessible from the command line.

Parameters

server

Either a domain name or an IP address expressed in dotted-decimal notation. If this optional field is omitted, `dig` attempts to use the default name server for your machine.

If you specify a domain name, `dig` resolves the query using the domain name resolver (BIND). If your system does not support DNS, you may have to specify an network address in dotted-decimal notation. Alternatively, if a DNS server is available, that server must be listed in the local hosts database.

domain

The domain name for which you are requesting information. See the `-x` option for a convenient way to specify a reverse translation address query.

query-type

The type of information (DNS query type) that you are requesting. If you omit this parameter, the default value for *query-type* is `a` (network address). BIND recognizes the following query types:

Query Type	Query Class	Description
`a`	T_A	Network address
`any`	T_ANY	All information about the specified domain
`mx`	T_MX	Mail exchanger for the domain
`ns`	T_NS	Name servers
`soa`	T_SOA	Zone of authority record
`hinfo`	T_HINFO	Host information
`axfr`	T_AXFR	Zone transfer (must ask an authoritative server)
`txt`	T_TXT	Arbitrary number of strings (see RFC 1035 for the complete list)

query-class

The network class requested in the query. If you omit this parameter, the default is `in` (C_IN, Internet class domain). BIND recognizes the following classes:

Query Type	Query Class	Description
`in`	C_IN	Internet class domain
`any`	C_ANY	All class information

See RFC 1035 for a complete list of query classes.

You can use the `query-class any` statement to specify a class or a type of query. `dig` parses the first occurrence of `any` to mean `query-type = T_ANY`. To specify `query-class = C_ANY`, you must either specify `any` twice or set `query-class` using the `-c` option.

Options

%ignored-comment

Use the percent (%) character to include an argument that is not parsed. This can be useful if you are running `dig` in batch mode. Instead of resolving every `@server-domain-name` in a list of queries, you can avoid the overhead of doing so, and still have the domain name on the command line as a reference. For example:

```
dig @128.9.0.32  %venera.isi.edu  mx  isi.edu
```

-<dig-option>

Use the hyphen (-) character to specify an option that affects the operation of `dig`. The options described in the Table A.1 are currently available (although not guaranteed to be useful). Options that are uppercase characters must be specified in quotes. For example, `dig -"P"`

Table A.1. dig Options

Option	Description

`-x` *`dot-notation-address`*	Convenient form for specifying reverse translation of IP address. Instead of:
	`dig 32.0.9.128.in-addr.arpa`
	you can use:
	`dig -x 128.9.0.32`

`-f` *`file`*	File for `dig` batch mode. The file contains a list of query specifications (`dig` command lines) that are to be executed successively. Lines beginning with ;, #, or `n` are ignored. Other options can still appear on the command line and will be in effect for each batch query.
`-T` *`time`*	Time (in seconds) between start of successive queries when running in batch mode. Can be used to keep two or more batch `dig` commands running synchronously. The default is 0.
`-p` *`port`*	Port number. Queries a name server listening to a nonstandard port number. The default is 53.
`-P`	After query returns, executes a `ping` command to compare response times. This option issues the following command: `$ MCR TCPIP$PING "-C" 3` *`server_name`* Use quotation marks to preserve the case of this option.

Option	Description
`-t query-type`	Type of query. Specifies either an integer value to be included in the type field, or uses the abbreviated mnemonic (such as `mx`).
`-c query-class`	Class of query. Specifies either an integer value to be included in the class field, or use the abbreviated mnemonic (such as `in`).

+<query-option>

Use the plus (+) character to specify an option to be changed in the query packet or to change `dig` output specifics. Many of these options are the same options accepted by `nslookup`. If an option requires a parameter, use the following format:

```
+ keyword [=value]
```

Most keywords can be abbreviated. Parsing of the "+" options is very simplistic– a value must not be separated from its keyword by any spaces. The following keywords are currently available:

Keyword	Abbreviation	Default	Description
`[no] debug`	`deb`	`deb`	Turn on/off debugging mode.
[no] `d2`		`nod2`	Turn on or off extra debugging mode.
[no] `recurse`	`rec`	`rec`	Use or do not use recursive lookup.
`retry=#`	`ret`	`4`	Set number of retries to #.
`time=#`	`ti`	`4`	Set timeout length to # seconds.
[no] `ko`		`noko`	Keep open option (implies `vc`).
[no] `vc`		`novc`	Use or do not use virtual circuit.
no `defname`	`def`	`def`	Use or do not use default domain name.
[no] `search`	`sea`	`sea`	Use or do not use domain search list.
`domain=NAME`	`do`		Set default domain name to NAME.
[no]`ignore`	`i`	`noi`	Ignore or do not ignore truncation errors.
[no] `primary`	`pr`	`nopr`	Use or do not use primary server.
no `aaonly`	`aa`	`noaa`	Authoritative query only flag.
[no] `cmd`		`cmd`	Echo parsed arguments.
[no] `stats`	`st`	`st`	Display query statistics.

Keyword	Abbreviation	Default	Description
[no] Header	H	H	Display basic header.
[no] header	he	he	Display header flags.
[no] ttlid	tt	tt	Display TTLs.
[no] cl		nocl	Display class information.
[no] qr		noqr	Display outgoing query
[no] reply	rep	rep	Display reply.
[no] ques	qu	qu	Display question section.
[no] answer	an	an	Display answer section.
[no] author	au	au	Display authoritative section.
[no] addit	ad	ad	Display additional section.
pfdef			Set to default display flags.
pfmin			Set to minimal default display flags.
pfset=#			Set display flags to # (# can be hexadecimal, octal, or decimal).
pfand=#			Bitwise and display flags with #.
pfor=#			Bitwise or display flags with #.

Examples

The following examples show how to use the `dig` command.

1. `$ dig`

```
; <<>> DiG 8.1 <<>>
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6
;; flags: qr rd ra; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13
;; QUERY SECTION:
;;      ., type = NS, class = IN

;; ANSWER SECTION:
.                       1d20h1m11s IN NS  E.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  D.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  A.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  H.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  C.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  G.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  F.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  B.ROOT-SERVERS.NET.
```

```
.                       1d20h1m11s IN NS  J.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  K.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  L.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  M.ROOT-SERVERS.NET.
.                       1d20h1m11s IN NS  I.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
E.ROOT-SERVERS.NET.     2d20h1m11s IN A  192.203.230.10
D.ROOT-SERVERS.NET.     2d20h1m11s IN A  128.8.10.90
A.ROOT-SERVERS.NET.     2d20h1m11s IN A  198.41.0.4
H.ROOT-SERVERS.NET.     2d20h1m11s IN A  128.63.2.53
C.ROOT-SERVERS.NET.     2d20h1m11s IN A  192.33.4.12
G.ROOT-SERVERS.NET.     2d20h1m11s IN A  192.112.36.4
F.ROOT-SERVERS.NET.     2d20h1m11s IN A  192.5.5.241
B.ROOT-SERVERS.NET.     2d20h1m11s IN A  128.9.0.107
J.ROOT-SERVERS.NET.     2d20h1m11s IN A  198.41.0.10
K.ROOT-SERVERS.NET.     2d20h1m11s IN A  193.0.14.129
L.ROOT-SERVERS.NET.     2d20h1m11s IN A  198.32.64.12
M.ROOT-SERVERS.NET.     2d20h1m11s IN A  202.12.27.33
I.ROOT-SERVERS.NET.     2d20h1m11s IN A  192.36.148.17

;; Total query time: 4013 msec
;; FROM: lassie.ucx.lkg.dec.com to SERVER: default -- 16.20.208.53
;; WHEN: Wed Aug  9 16:42:08 2000
;; MSG SIZE  sent: 17  rcvd: 436
```

This example shows how to query your default name server for query type NS (default query type) and query class IN (default query class). The output shows the address records for the root name servers and their IP addresses.

2. `$ dig microsoft.com mx`

```
; <<>> DiG 8.1 <<>> microsoft.com mx
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 12, ADDITIONAL: 8
;; QUERY SECTION:
;;      microsoft.com, type = MX, class = IN

;; ANSWER SECTION:
microsoft.com.          2h18m8s IN MX   10 mail2.microsoft.com.
microsoft.com.          2h18m8s IN MX   10 mail3.microsoft.com.
microsoft.com.          2h18m8s IN MX   10 mail4.microsoft.com.
microsoft.com.          2h18m8s IN MX   10 mail5.microsoft.com.
microsoft.com.          2h18m8s IN MX   10 mail1.microsoft.com.

;; AUTHORITY SECTION:
com.                    5d22h12m9s IN NS  A.ROOT-SERVERS.NET.
com.                    5d22h12m9s IN NS  E.GTLD-SERVERS.NET.
com.                    5d22h12m9s IN NS  F.GTLD-SERVERS.NET.
com.                    5d22h12m9s IN NS  F.ROOT-SERVERS.NET.
com.                    5d22h12m9s IN NS  J.GTLD-SERVERS.NET.
com.                    5d22h12m9s IN NS  K.GTLD-SERVERS.NET.
com.                    5d22h12m9s IN NS  A.GTLD-SERVERS.NET.
com.                    5d22h12m9s IN NS  M.GTLD-SERVERS.NET.
com.                    5d22h12m9s IN NS  G.GTLD-SERVERS.NET.
com.                    5d22h12m9s IN NS  C.GTLD-SERVERS.NET.
```

```
com.                    5d22h12m9s IN NS  I.GTLD-SERVERS.NET.
com.                    5d22h12m9s IN NS  B.GTLD-SERVERS.NET.

;; ADDITIONAL SECTION:
mail2.microsoft.com.    2h8m41s IN A    131.107.3.124
mail3.microsoft.com.    2h27s IN A      131.107.3.123
mail4.microsoft.com.    1h53m4s IN A    131.107.3.122
mail5.microsoft.com.    2h8m43s IN A    131.107.3.121
mail1.microsoft.com.    2h8m43s IN A    131.107.3.125
A.ROOT-SERVERS.NET.     2d19h47m37s IN A  198.41.0.4
E.GTLD-SERVERS.NET.     1d9h45m57s IN A  207.200.81.69
F.GTLD-SERVERS.NET.     3h16m16s IN A   198.17.208.67

;; Total query time: 4019 msec
;; FROM: lassie.ucx.lkg.dec.com to SERVER: default -- 16.20.208.53
;; WHEN: Wed Aug  9 16:55:42 2000
;; MSG SIZE  sent: 31  rcvd: 493
```

This example shows how to obtain the mail server records for Microsoft.

ifconfig

ifconfig — Assigns an address to a network interface, and configures and displays network interface parameters.

Syntax

`ifconfig` *interface_id* [*address_family*] [*address*[*/bitmask*]] [*dest_address*] [*parameters*])

`ifconfig -a [-d] [-u] [-v] [`*address_family*`]`

`ifconfig -l [-d] [-u] [-v] [`*address_family*`]`

`ifconfig [-v]` *interface-id* [*address_family*]

Description

Use the `ifconfig` command to define the network address of each interface. You can also use the `ifconfig` command at other times to display all interfaces that are configured on a system, to redefine the address of an interface, or to set other operating parameters.

Note

If you want to redefine the interface address or the net mask, you should stop VSI TCP/IP Services. Otherwise, any TCP/IP process currently running will continue to use the old address and net mask and will fail.

Any user can query the status of a network interface; only a privileged user can modify the configuration of network interfaces.

You specify an interface with the `ifconfig` *interface-id* syntax. (See your hardware documentation for information on obtaining an interface ID.)

If you specify only *interface-id*, the `ifconfig` program displays the current configuration for the specified network interface only.

If a protocol family is specified by the *address_family* parameter, `ifconfig` reports only the configuration details specific to that protocol family.

When changing an interface configuration, if the address family cannot be inferred from the *address* parameter, an address family must be specified. The address family is required because an interface can receive transmissions in different protocols, each of which can require a separate naming scheme.

The *address* parameter is the network address of the interface being configured. For the `inet` address family, the *address* parameter is either a host name or an IP address in the standard dotted-decimal notation with or without the optional Classless Inter-Domain Routing (CIDR) bit mask (*/bitmask*). If you specify *bitmask*, do not use the *netmask* parameter.

The destination address (*dest_address*) parameter specifies the address of the correspondent on the remote end of a point-to-point link.

Flags

-a

Displays information about all interfaces that are configured on a system.

-d

Displays information about interfaces that are down.

-l

Displays interface names that are configured on a system.

-u

Displays information about interfaces that are up.

-v

Displays detailed information about interfaces, such as hardware addresses.

Parameters

alias *alias_address*[*/bitmask*]

Establishes an additional network address for this interface. This can be useful when changing network numbers and you want to continue to accept packets addressed to the old interface.

If you do not specify a bit mask or net mask with the alias address, the default net mask is based on the alias address's network class.

If you are using the optional bit mask argument, do not use the net mask argument.

-alias *alias_address*

Removes the network address specified. This can be used if you incorrectly specified an alias or if an alias is no longer needed. The `-alias` parameter functions in the same manner as the delete parameter.

aliaslist *address_list*[/*bitmask*]

Establishes a range of additional network addresses for this interface. The range can be a comma-separated list or a hyphenated list, and is inclusive. You can also specify the optional CIDR bit mask (/*bitmask*) argument at the end of the list. Do not use both a comma-separated list and a hyphenated list for a range.

-aliaslist

Removes a range of network addresses for this interface. This can be useful when deleting network numbers and you want to keep the primary interface address. The `-aliaslist` rules are the same as for the `aliaslist` parameter.

allmulti

Enables the reception of all multicast packets.

-allmulti

Disables the reception of all multicast packets.

arp

Enables the use of the Address Resolution Protocol (ARP) in mapping between network-level addresses and link-level addresses. This parameter is on by default.

-arp

Disables the use of the ARP. Use of this parameter is not recommended.

broadcast *broad_address*

Specifies the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part consisting of all ones (1). Note that the computation of the host part is dependent on `netmask` (see the description of the *netmask* parameter).

delete [*net_address*]

Removes the network address specified. Use this parameter if you incorrectly specified an alias or if the alias was no longer needed. If you have incorrectly set an NS address, removing all NS addresses w ill allow you to specify the host portion again.

If no address is specified, `ifconfig` deletes all network addresses for the interface.

down

Marks an interface as not working (down), which keeps the system from trying to transmit messages through that interface. If possible, the `ifconfig` command also resets the interface to disable the reception of messages. Routes that use the interface, however, are not automatically disabled.

[-]fail

Controls the specified interface. You can force the interface to fail by specifying `fail` and to recover by specifying `-fail`.

[-]home

Forces an alias address to be created with a home interface. This option is useful when you are creating an IP address. By default, all primary IP addresses are created with a home interface.

[-]fs

Creates an IP address that is not managed by failSAFE IP. By default, if the failSAFE IP services is running, all IP addresses are created as failSAFE IP addresses except for those addresses assigned to the loopback interface (LO0) (for example, the localhost address 127.0.0.1.

ipmtu ***mtu_value***

Alters the size of the maximum transmission unit (MTU) for messages that your system transmits. It might be necessary to reduce the MTU size so that bridges connecting token rings can transfer frames without error.

metric ***number***

Sets the routing metric, or number of hops, for the interface to the value of *number*. The default value is zero (0) if *number* is not specified, indicating that both hosts are on the same network. The routing metric is used by the ROUTED and GATED services, with higher metrics indicating that the route is less favorable.

netmask ***mask***

Specifies how much of the address to reserve for subdividing networks into sub-networks. This parameter can only be used with an address family of `inet`. Do not use this parameter if you are specifying the CIDR mask (*/bitmask*) with the *address* parameter, *alias* parameter, or *aliaslist* parameter.

The `mask` variable includes both the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number beginning with 0x, in the standard Internet dotted-decimal notation, or beginning with a name.

The mask contains ones (1) for the bit positions in the 32-bit address that are reserved for the network and subnet parts, and zeros (0) for the bit positions that specify the host. The mask should contain at least the standard network portion.

The default net mask is based on the address parameter's network class.

up

Marks an interface as working (up). This parameter is used automatically when setting the first address for an interface, or it can be used to enable an interface after an `ifconfig down` command. If the interface was reset when previously marked with the `down` parameter, the hardware will be reinitialized.

Examples

The following examples show how to use the `ifconfig` command.

1.
```
TCPIP>  ifconfig sl0
sl0: flags=10
```

   This example shows how to query the status of serial line interface sl0.

2. `TCPIP>  ifconfig lo0 inet  127.0.0.1 up`

   This example shows how to configure the local loopback interface. Only a user with system privileges can modify the configuration of a network interface.

3. `TCPIP>  ifconfig ln0 212.232.32.1/22`

   This example shows how to configure an ln0 interface. The broadcast address is 212.232.35.255 as the 22-bit mask specifies four Class C networks.

4. `TCPIP>  ifconfig tra0 130.180.4.1/24 speed 4`

   This example shows how to configure the token ring interface for a 4 Mb/s token ring with a net mask of 255.255.255.0 in CIDR format.

5. ```
   TCPIP>  ifconfig tra0 down
   TCPIP>  ifconfig tra0 speed 16 up
   ```

   This example shows how to stop the token ring interface and start it for a 16 Mb/s token ring.

6. `TCPIP>  ifconfig we0 alias 132.50.40.35/24`

   This example shows how to add alias 132.50.40.35 with a net mask of 255.255.255.0 in CIDR format to interface we0.

7. `TCPIP>  ifconfig we0 aliaslist 132.240.32-36.40-50/24`

   This example shows how to add network addresses 40 through 50, to subnets 18.240.32, 18.240.33, 18.240.34, 18.240.35, and 18.240.36 with a net mask of 255.255.255.0 in CIDR format to the we0 interface.

8. ```
   TCPIP>  ifconfig we0 down delete abort
   145.92.16.1: aborting 7 tcp connection(s)
   ```

   This example shows how to stop Ethernet interface we0, delete all addresses associated with the interface, and close all TCP connections.

9. ```
   TCPIP>  ifconfig we0 -alias 145.92.16.2 abort
   145.92.16.2: aborting 2 tcp connection(s)
   ```

   This example shows how to delete the alias address 145.92.16.2 on interface tu0 and close all TCP connections.

10. `TCPIP>  ifconfig we0 alias 145.92.16.2 physaddr aa:01:81:43:02:11`

    This example shows how to associate MAC address aa:01:81:43:02:11 with the alias address 145.92.16.2.

11. `TCPIP>  ifconfig we0 -alias 145.92.16.2 -physaddr aa:01:81:43:02:11`

    This example shows how to disassociate MAC address aa:01:81:43:02:11 from the alias address 145.92.16.2.

12. ```
    A72KT: ifconfig -l
    TCPIP>ifconfig -l
    LO0 TN0 WE0
    ```

    This example shows how to display the names of the interfaces on the system only.

13.
```
TCPIP> ifconfig -v WE0
wE0: flags=c43<UP,BROADCAST,RUNNING,MULTICAST,SIMPLEX>
     HWaddr aa:0:4:0:72:f8
     inet 10.10.2.2 netmask ffffff00 broadcast 10.10.2.255 ipmtu 1500
```

This example shows how to display the hardware and IP addresses of interface WE0.

ndc

ndc — Manages the BIND server.

Syntax

`ndc` *`directive`* `[`*`directive`* `... ]`

Description

This command allows the name server administrator to send various messages to a name server. You can specify zero or more directives from the following list.

Directives

status

Displays the current status of the BIND server process.

dumpdb

Causes the BIND server to dump its database and cache to SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND_SERVER_ZONES_DUMP.DB.

reload

Causes the BIND server to check the serial numbers of all primary and secondary zones and to reload those that have changed.

stats

Causes the BIND server to dump statistics to SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND_SERVER_STATISTICS.LOG.

trace

Causes the BIND server to increment its tracing level by 0. Trace information is written to SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND_RUN.LOG. Higher tracing levels result in more detailed information.

notrace

Causes the BIND server to set its tracing level to 0.

start

Causes the BIND server to be started, if it is not running.

stop

Causes the BIND server to be stopped if it is running.

restart

Causes the BIND server to be stopped and restarted.

Examples

The following examples show how to use the `ndc` command.

1.
```
$ ndc status
BIND Server process information:
  Process ID:                   44C0021C
  Process name:               TCPIP$BIND_1
  Priority:                              9
  Elapsed CPU time:        0 00:00:31.19
  Buffered I/O count:             214082
  Direct I/O count:                  404
  Page Faults:                       485
  Pages:                            4096
  Peak virtual size:              173696
  Peak working set size:            5920
  Process state:                     LEF
```

   This example shows how to display the current status of the BIND server process.

2.
```
$ ndc dumpdb
```

   This example shows how to dump the BIND server's database into the SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND_SERVER_ZONES_DUMP.DB file. Use the DCL command TYPE to view the contents of this file.

3.
```
$ ndc stats
```

   This example shows how to dump BIND server statistics to the SYS$SPECIFIC:[TCPIP$BIND]TCPIP$BIND_SERVER_ZONES_STATISTICS.LOG file. Use the DCL command TYPE to view the contents of this file.

netstat

netstat — Displays network-related data in various formats.

Syntax

```
netstat [-rn] | [-an] [-f address_family] [interval] ]

netstat [-abd"H"im"M"nrstv] [-f address_family] [interval]

netstat [-ntdz] ["-I" interface] [interval]

netstat [-i] [-p protocol]
```

Description

The interval argument specifies in seconds the interval for updating and displaying information. The first line of the display shows cumulative statistics; subsequent lines show statistics recorded during interval.

Default display

When used without flags, the `netstat` command displays a list of active sockets for each protocol. The default display shows the following items:

- Local and remote addresses
- Send and receive queue sizes (in bytes)
- Protocol
- State

Address formats are of the form *host.port* or *network.port* if a socket's address specifies a network but no specific host address. The host and network address are displayed symbolically unless `-n` is specified.

Interface display

The network interface display provides a table of cumulative statistics for the following:

- Interface name (Name)
- Maximum transmission unit (Mtu)
- Network address
- Packets received (Ipkts)
- Packets received in error (Ierrs)
- Packets transferred (Opkts)
- Outgoing packets in error (Oerrs)
- Collisions (Coll)

  Note that the collisions item has different meanings for different network interfaces.
- Drops (optional with `-d`)
- Timers (optional with `-t`)

Routing table display

A route consists of a destination host or network and a gateway to use when forwarding packets.
Direct routes are created automatically for each interface attached to the local host when you issue the `ifconfig` command. Routes can be modified automatically in response to the prevailing condition of the network.

The routing-table display format indicates available routes and the status of each in the following fields:

Flags

Displays the state of the route as one or more of the following:

U	Up, or available.
G	This route is to a gateway.
H	This route is to a host.
D	This route was dynamically created by a redirect.
M	This route was modified by a redirect.
S	This is a static route that was created by the `route` command.
R	This is a reject route that was created by the `route` command.

refcnt

Gives the current number of active uses for the route. Connection-oriented protocols hold on to a single route for the duration of a connection; connectionless protocols obtain routes in the process of sending to a destination.

use

Provides a count of the number of packets sent using the route.

interface

Indicates the network interface used for the route.

When the `-v` flag is specified, the routing table display includes the route metrics. An asterisk (*) indicates the metric is locked.

Flags

-a

Displays the state of sockets related to the Internet protocol. Includes sockets for processes such as servers that are currently listening at a socket but are otherwise inactive.

-b

Displays the contents of the Mobile IPv6 binding cache. When used with the -s option, it displays binding cache statistics.

For more information about using netstat with IPv6, refer to the *VSI TCP/IP Services for OpenVMS Guide to IPv6*.

-d

Displays the number of dropped packets; for use with the `-I` interface or `-i` flags. You can also specify an interval argument (in seconds).

-f *address_family*

Limits reports to the specified address family. The address families that can be specified might include the following:

inet	Specifies reports of the AF_INET family, if present in the kernel.
all	Lists information about all address families in the system.
any	Lists information about any address families in the system.

-H

Displays the current ARP table (behaves like `arp -a.`)

Use quotation marks to preserve the case of this option.

-i

Displays the state of configured interfaces. (Interfaces that are statically configured into the system but not located at system startup are not shown.)

When used with the `-a` flag, `-i` displays IP and link-level addresses associated with the interfaces.

You can use the `-i` flag to retrieve your system's hardware address.

You can use the -p flag to specify a protocol for which to display statistics.

-I *interface*

Displays information about the specified interface.

-p protocol

Displays statistics for the specified protocol. Use the `-i` flag with the `-p` flag.

-m

Displays information about memory allocated to data structures associated with network operations.

"-M"

Displays Internet protocol multicast routing information. When used with the `-s` flag, `-M` displays IP multicast statistics.

-n

Displays network address in numerical format with network masks in CIDR format. When this flag is not specified, the address is displayed as host name and port number. This flag can be used with any of the display formats.

-r

Displays the host's routing tables. When used with the `-s` flag, `-r` shows the host's routing statistics instead of its routing tables.

-s

Displays statistics for each protocol.

-t

Displays timer information. Use with the `-I` interface or `-i` flag.

-v

Displays detailed output when specified with the `-r` flag. In this case, route metric values are displayed.

-z

Sets the network interface counters to zero. This flag must be specified with the `-I` interface flag. You must have system privileges to use the `-z` flag.

Examples

The following examples show how to use the `netstat` command to display information about configured interfaces and routing tables.

1.
```
TCPIP> netstat -i
Name  Mtu   Network   Address           Ipkts Ierrs    Opkts Oerrs Coll
LO0   4096  <Link>    Link#1              167     0      167     0    0
LO0   4096  loop      LOCALHOST           167     0      167     0    0
SE0   1500  <Link>    aa:0:4:0:6d:f8     1544     0      157     0    0
SE0   1500  loop      RUFUS              1544     0      157     0    0
TN0*  1280  <Link>    Link#2                0     0        0     0    0
```

   This example shows how to display the state of the configured interfaces.

2.
```
TCPIP> netstat -r
Routing tables
Destination      Gateway            Flags     Refs     Use  Interface

Route Tree for Protocol Family 26:

Route Tree for Protocol Family 2:
default          sqagate            UG          0        0  SE0
10.10.2          v72kt              U           2      125  SE0
v72kt            v72kt              UHL         0       50  SE0
LOCALHOST        LOCALHOST          UHL         7      117  LO0
```

   This example shows how to display the routing tables.

3.
```
TCPIP> netstat -rn
Routing tables
Destination      Gateway            Flags     Refs     Use  Interface

Route Tree for Protocol Family 26:

Route Tree for Protocol Family 2:
default          10.10.2.66         UG          0        0  SE0
10.10.2/24       10.10.2.4          U           2      109  SE0
10.10.2.4        10.10.2.4          UHL         0       50  SE0
127.0.0.1        127.0.0.1          UHL         7      117  LO0
```

This example shows how to display the routing tables with network addresses.

nslookup

nslookup — Queries Internet name servers interactively.

Syntax

`nslookup [-option ...] [host_to_find | - [server] ]`

Description

The `nslookup` command is a program that is used to query Internet domain name servers. The `nslookup` command has two modes: noninteractive and interactive.

Note

The `nslookup` utility is deprecated. VSI recommends that you use the `dig` utility instead. For information about the dig utility, see the *VSI TCP/IP Services for OpenVMS Management* guide.

Noninteractive mode

Noninteractive mode is used to display just the name and requested information for a host or domain. Noninteractive mode is invoked when the name or Internet address of the host to be looked up is given as the first argument. The optional second argument specifies the host name or address of a name server.

Interactive mode

Interactive mode allows the user to query name servers for information about various hosts and domains or to display a list of hosts in a domain. Interactive mode is invoked when you specify `nslookup` without arguments (the default name server will be used), or when the first argument you specify is a hyphen (-) and the second argument is the host name or IP address of a name server.

The options listed under the `set` command can be specified in the `.nslookuprc` file in the user's home directory if they are listed one per line. Options can also be specified on the command line if they precede the arguments and are prefixed with a hyphen (-). For example, to change the default query type to host information, and the initial time out to 10 seconds, enter the following command:

```
$ nslookup -query=hinfo  -timeout=10
```

Interactive commands

Commands can be interrupted at any time by pressing Ctrl/C. To exit, press Ctrl/D (EOF) or type `exit`. The command line length must be less than 256 characters. To treat a built-in command as a host name, prefix it with an escape character (^) plus a backslash (\). Note that an unrecognized command will be interpreted as a host name.

Commands

***host* [*server*]**

Looks up information for the host using either the current default server or the specified server. If `host` is an IP address and the query type is A or PTR, the name of the host is returned. If `host`

is a name and does not have a trailing period, the default domain name is appended to the name. (This behavior depends on the state of the `set` options `domain`, `srchlist`, `defname`, and `search`.) To look up a host not in the current domain, append a dot (.) to the end of the domain name.

server *domain*
lserver *domain*

Changes the default server to *`domain`*. The `lserver` command uses the initial server to look up information about *`domain`*, while the `server` command uses the current default server. If an authoritative answer cannot be found, the names of servers that might have the answer are returned.

root

Changes the default server to the server for the root of the domain name space. Currently, the host `ns.internic.net` is used. (This command is a synonym for `lserver ns.internic.net.`) The name of the root server can be changed with the `set root` command.

finger [*name*] [> *filename*]
finger [*name*] [>> *filename*]

Connects with the `finger` server on the current host. The current host is defined when a previous lookup for a host was successful and returned address information (see the `set querytype=A` command). The redirection symbols (> and >>) can be used to redirect output in the usual manner.

ls [*option*] *domain* [> *filename*]
ls [*option*] *domain* [>> *filename*]

Lists the information available for *domain*, optionally creating or appending to *filename*. The default output contains host names and their IP addresses. The value for `option` can be one of the following:

Option	Description
`-t` *querytype*	Lists all records of the specified type. (See `querytype` in Table A.2.)
`-a`	Lists aliases of hosts in the domain. This option is a synonym for `-t CNAME`.
`-d`	Lists all records for the domain. This option is a synonym for `-t ANY`.
`-h`	Lists CPU and operating system information for the domain. This option is a synonym for `-t HINFO`.
`-s`	Lists well-known services of hosts in the domain. This option is a synonym for `-t WKS`. When output is directed to a file, a pound sign (#) is displayed for every 50 records received from the server.

view *filename*

Sorts and lists the output of previous `ls` commands.

help

Displays a brief summary of commands.

exit

Exits the program.

set *keyword* [=*value*]

Use the command to change state information that affects the lookups. Table A.2 lists the valid keywords.

Table A.2. Options to the `nslookup set` Command

Keyword	Function	
`ALL`	Displays the current values of the options you can set as well as information about the current default server. For example: `> set all`	
`class=value`	Changes the query class to one of the following: • `IN` — The internet class (default) • `CHAOS` — The chaos class • `ANY` — Wildcard The class specifies the protocol group of the information. You abbreviate this keyword to `cl`. This command tells `nslookup` to resolve both `in` and `chaos` class queries (you can enter `in` and `chaos`): `> set class=ANY`	

`querytype`	Specifies the type of information you want. For example:
	`> set querytype=A`
	`> set querytype=ANY`
	Valid types are:	

	SOA	Start of authority. Marks the beginning of a zone's data and defines parameters that affect the entire zone.
	NS	Name server. Identifies a domain's name server.
	A	Address. Maps a host name to an address.
	ANY	Defines all available resource records for a given name.
	PTR	Pointer. Maps an address to a host name.

Keyword	Function	
	MX	Identifies where to deliver mail for a given domain.
	CNAME	Defines an alias host name.
	HINFO	Host information. Describes a host's hardware and operating system.
	WKS	Well-known service. Advertises network services.
`[no]debug`	Turns on debugging (default is `nodebug`). `nslookup` displays detailed information about the packet sent to the server and the answer. For example: `> set debug` You can use the abbreviations `nodeb` and `deb`.	
`[no]d2`	Returns all-inclusive debugging information (default is `nod2`). Displays all the fields of every packet. For example: `> set d2`	
`recurse`	Tells the BIND server to contact other servers if it does not have the information you want. The servers carry out a complete (recursive) resolution for each query. For example: `> set recurse`	
`retry`	Number of times that `nslookup` attempts to contact a BIND server if repeated tries fail. For example: `> set retry=8`	
`timeout`	Length of time to wait for a reply from each attempt. For example: `> set timeout=9`	
`root=`*`value`*	Changes the root server. For example, the following command changes the root server to `ns.nasa.gov`. `> set root=ns.nasa.gov`	
`ignoretc`	Tells `nslookup` to ignore packet truncation errors. For example: `> set ignoretc`	
`domain` *`name`*	Changes the default domain to the domain you specify. The settings of the `defname` and `search` options control how the default domain name is appended to lookup requests. The domain search list contains the parents of the default domain if the default domain has at least two components in its name. The default value is set in the TCP/IP configuration database. To specify the default, type the abbreviation `do`. For example, if the default domain is `CC.Berkeley.EDU`, the search list is `CC.Berkeley.EDU` and `Berkeley.EDU`.	

Keyword	Function
`srchlist`	If set, `nslookup` appends each of the domain names specified in the `srchlist` option to an unqualified host name and performs a query until an answer is received.
`srchlist=`*`names`*	Changes the default domain name to the first name you specify, and changes the domain search list to all the names you specify. Specify a maximum of six names separated by slashes (/). In the following example, the command sets the default domain to `lcs.MIT.EDU` and changes the search list to the three specified domains. The command overrides the default domain name and associated search list for the `set domain` command. `> set srchlist=lcs.MIT.EDU/ai.MIT.EDU/MIT.EDU` The default is the domain name specified in the TCP/IP configuration database. The abbreviated form of the command is `srchl`.
`[no]defname`	Tells `nslookup` to append a default domain name to a lookup request if the specified DNS name is not fully qualified.[1] The abbreviated form is `[no]def`. For example, an `nslookup` query for the host `rainy` becomes `rainy.cc.berkeley.edu`.
`[no]search`	Tells `nslookup` to append the search list domain names to the lookup request domain name if the lookup request domain name is not fully qualified.[1] The default is `search`. The abbreviated form is `[no]sea`.

[1] A fully qualified domain name is a name that ends with a dot (.), as in *host.domain.*

Examples

The following example shows how to use `nslookup` interactively.

1.
```
$ nslookup
Default Server:  condor.lgk.dec.com
Address:  16.99.208.53

> set all
Default Server:  condor.lgk.dec.com
Address:  16.99.208.53

Set options:
  nodebug             defname         search          recurse
  nod2                novc            noignoretc      port=53
  querytype=A         class=IN        timeout=4       retry=4
  root=a.root-servers.net.
  domain=xyz.prq.dec.com
  srchlist=xyz.prq.dec.com
```

ping

ping — Send ICMP ECHO_REQUEST packets to network hosts.

Syntax

`ping [-dfnqruvR] -c count [-i wait] [-l preload] [-p pattern] [-s packetsize] host`

Description

The `ping` command uses the ICMP (Internet Control Message Protocol) mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE message from the specified host or gateway host. ECHO_REQUEST datagrams (`ping`) have an IP (Internet Protocol) and ICMP header, followed by a `struct timeval` and then an arbitrary number of pad bytes used to fill out the packet.

When using `ping` for fault isolation, first run the command on the local host to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be sent the `ping` command. Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculations, although the round-trip time of these packets is used in calculating the minimum, average, and maximum round-trip time numbers. When the specified number of packets have been sent (and received), or if the program is terminated with a SIGINT, a brief summary is displayed.

This program is intended for use in network testing, measurement, and management. Because of the load it can impose on the network, it is unwise to use `ping` during normal operations or from automated scripts.

ICMP packet details

An IP header without options is 20 bytes. An ICMP ECHO_REQUEST packet contains an additional 8 bytes worth of ICMP header followed by an arbitrary amount of data. When a packetsize is given, this indicates the size of this extra piece of data (the default is 56). Thus, the amount of data received inside of an IP packet of type ICMP ECHO_REPLY will always be 8 bytes more than the requested data space (the ICMP header).

If the data space is at least 8 bytes large, `ping` uses the first 8 bytes of this space to include a timestamp, which it uses in the computation of round-trip times. If less than 8 bytes of pad are specified, no round-trip times are given.

Duplicate and damaged packets

The `ping` command will report duplicate and damaged packets. Duplicate packets should never occur, and seem to be caused by inappropriate link-level retransmissions. Duplicates can occur in many situations and are rarely (if ever) a good sign, although the presence of low levels of duplicates can not always be cause for alarm.

Damaged packets are obviously serious cause for alarm and often indicate broken hardware somewhere in the `ping` packet's path (in the network or in the hosts).

Different data patterns

The network layer should never treat packets differently depending on the data contained in the data portion. Unfortunately, data-dependent problems have been known to invade networks and remain undetected for long periods of time. In many cases the problematic pattern does not have sufficient transitions, such as all ones (1) or all zeros (0), or has a pattern at the right, such as almost all zeros (0). It is not necessarily enough to specify a data pattern of all zeros on the command line because the problematic pattern of interest is at the data-link level, and the relationship between what you enter and what the controllers transmit can be complicated.

Data-dependent problems can be identified only by extensive testing. If you are lucky, you can manage to find a file that either cannot be sent across your network or that takes much longer to transfer than other files of similar length. You can then examine this file for repeated patterns that you can test by using the `-p` option to the `ping` command.

TTL details

The TTL value of an IP packet represents the maximum number of IP routers that the packet can go through before being thrown away. You can expect each router in the Internet to decrement the TTL field by exactly one.

The maximum possible value of this field is 255, and most UNIX compatible systems set the TTL field of ICMP ECHO_REQUEST packets to 255. This is why you can use the `ping` command on some hosts but not reach them with `TELNET` or `FTP`.

In normal operation, `ping` displays the TTL value from the packet it receives. When a remote system receives a `ping` packet, it can do one of three things with the TTL field in response:

- Not change the field. This is what Berkeley UNIX compatible systems did before BSD Version 4.3. In this case, the TTL value in the received packet will be 255 minus the number of routers in the round-trip path.
- Set the field to 255. This is what current Berkeley UNIX compatible systems do. In this case, the TTL value in the received packet will be 255 minus the number of routers in the path from the remote system to the host that received the `ping` commands.
- Set the field to some other value. Some machines use the same value for ICMP packets that they use for TCP packets; for example, either 30 or 60. Others may use completely wild values.

Cautions

Many hosts and gateways ignore the RECORD_ROUTE option.

Flooding and preloading the `ping` command is generally not recommended, and flooding `ping` messages on the broadcast address should be done only under very controlled conditions.

Flags

-c *count*

Stops after sending (and receiving) the specified number (*count*) of ECHO_RESPONSE packets.

-d

Sets the `SO_DEBUG` option on the socket being used.

-f

Floods `ping`. Outputs packets as fast as they come back or 100 times per second, whichever is more. For every `ECHO_REQUEST` sent, a dot (.) is displayed, while for every `ECHO_REPLY` received a backspace is used. This provides a rapid display of how many packets are being dropped. You must have system privileges to use this option. Using the `-f` flag can be very hard on a network and should be used with caution.

-i *wait*

Waits the specified number of seconds between sending each packet. The default is to wait for 1 second between each packet. This option is incompatible with the `-f` option.

-l *preload*

If preload is specified, `ping` sends that many packets as fast as possible before falling into its normal mode of behavior. You must have system privileges to use this option. Using the `-l` option can be very hard on a network and should be used with caution.

-n

Numeric output only. No attempt is made to look up symbolic names for host addresses. This occurs only when displaying ICMP packets other than ECHO_RESPONSE.

-p *pattern*

Specifies up to 16 pad bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, `-p ff` will cause the sent packet to be filled with all ones (1).

-q

Suppresses output. Nothing is displayed except the summary lines at startup time and at completion.

-R

Records route. Includes the RECORD_ROUTE option in the ECHO_REQUEST packet and displays the route buffer on returned packets. Note that the IP header is large enough for only nine such routes. Many hosts ignore or discard this option.

-r

Bypasses the normal routing tables and sends directly to a host on an attached network. If the host is not on a directly attached network, an error is returned. This option can be used to send `ping` to a local host through an interface that has no route through it (for example, after the interface was dropped by ROUTED).

-s *packetsize*

Specifies the number of data bytes to be sent. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data.

-u

Displays the time in microseconds (three decimal places). In order to ensure this microsecond precision, the NTP_TIME and MICRO_TIME kernel options must be on. By default, NTP_TIME and MICRO_TIME kernel options are off. If these kernel options are off and this flag is used, the time is displayed to three decimal places, but in milliseconds.

-v

Specifies detailed output. ICMP packets other than ECHO_RESPONSE that are received are listed.

Examples

The following example shows how to use the `ping` command.

1.
```
TCPIP> ping
PING rufus.lkg.dec.com (10.10.2.4): 56 data bytes
64 bytes from 10.10.2.4: icmp_seq=0 ttl=64 time=30 ms
64 bytes from 10.10.2.4: icmp_seq=1 ttl=64 time=0 ms
64 bytes from 10.10.2.4: icmp_seq=2 ttl=64 time=0 ms
64 bytes from 10.10.2.4: icmp_seq=3 ttl=64 time=0 ms


----rufus.lkg.dec.com PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/8/30 ms
```

route

route — Manually manipulates the routing tables.

Syntax

```
route [-nqvC] add [-net | -host] [family] destination[/bitmask]
gateway [-link] [modifiers [args]]

route [-nqv] change [-net | -host] [family] destination gateway [-
link] [modifiers args]

route [-n] monitor

route [-nqvC] delete [-net | -host] [family] destination[/bitmask]
[-link] gateway [modifiers args]

route [-nqvC] flush [family]
```

Description

The `route` command allows you to manually manipulate the network routing tables. It normally is not needed, since a system-routing table management daemon, such as GATED or ROUTED, should tend to this task.

The `route` command accepts five commands:

`add`	Adds a route.
`flush`	Removes all routes.
`delete`	Deletes a specific route.
`change`	Changes aspects of a route (such as its gateway).

`monitor`	Reports any changes to the routing information base, r' lookup misses, or suspected network partitions.

The `flush` command has the following format:

```
route [-n] flush [family]
```

In this format, the address family can be specified by the `-inet` keyword.

The other commands have the following format:

```
route [-n]  command [-net | -host] destination[/bitmask] gateway modifier
 [-netmask mask]
```

Unless the *net* or the *host* parameter is specified on the command line, route creates a host route or a network route by interpreting the Internet address associated with destination parameter. If the destination has a local address part of INADDR_ANY, or if the destination is the symbolic name of a network, a network route is created; otherwise, a host route is created.

For example, `128.32` is interpreted as `-host 128.0.0.32`, 128.32.130 is interpreted as `-host 128.32.0.130`; `-net 128.32` is interpreted as `128.32.0.0`, and `-net 128.32.130` is interpreted as `128.32.130.0`.

All symbolic names specified for a destination or gateway are looked up first as a host name using `gethostbyname()`. If this lookup fails, `getnetbyname()` is then used to interpret the name as that of a network.

Routes added with the route command are marked as RTF_STATIC to differentiate them from routes added by the routing daemons (GATED or ROUTED). The GATED daemon does not remove the RTF_STATIC routes when it is shut down.

The `route` utility uses a routing socket and the new message types RTM_ADD, RTM_DELETE, and RTM_CHANGE. As such, only a privileged user can modify the routing tables.

Flags

-n

Prevents attempts to display host and network names symbolically when reporting actions.

-v

Displays additional details.

-q

Suppresses all output.

Modifiers

-all

Specifies that the kernel add or delete the specified route on all interfaces (for example, `TU0` and `TU1`) that are in the same subnet as the gateway. Use this modifier only with the `add` and `delete` modifiers. Do not use `-all` with the `-dev` and `-olddev` modifiers.

-blackhole

Specifies that this route is a blackhole route. Packets sent to blackhole routes are dropped, and notification is not sent to the packet originators. This is different from a normal route, which allows packets to be forwarded out on it. You must specify `127.0.0.1` (localhost) as the gateway argument.

-cloning

Generates a new route on use of this route.

-dev *device*

Specifies the interface device to use in the routing entry. Use this modifier when you want to designate a particular interface for a route. If you do not specify this modifier, the route is added on the first interface that is found.

-genmask *mask*

Specifies that the netmask *mask* is used for all routes cloned from this route.

-hopcount *count*

Sets this route's maximum hopcount to the value specified by *count*.

-iface | -interface

Specifies that this route is through an interface instead of through a gateway (gateway is the default). This means the destination is reachable directly through an interface; no intermediate system is required. The gateway parameter is the host address to be used for transmission.

-inet

Sets this route's type as AF_INET. When you specify `inet` with the `delete` or `flush` command, only AF_INET routes are deleted.

-llinfo

Specifies that this route contains valid link-layer information.

-lock

Locks the metric specified by the next modifier specified on the command line. A locked metric is not modified by the kernel. The following metrics can be locked: `mtu`, `hopcount`, `recvpipe`, `sendpipe`, `ssthresh`, `rtt`, and `rttvar`.

-lockrest

Locks the metrics specified by all the modifiers that follow on the command line. A locked metric is not modified by the kernel. The following metrics can be locked: `mtu`, `hopcount`, `recvpipe`, `sendpipe`, `ssthresh`, `rtt`, and `rttvar`.

-mtu *size*

Sets this route's maximum transmission unit (MTU) (in bytes) to the value specified by *size*.

-netmask *mask*

Specifies the subnet mask to use for the routing entry. Networks that use a nonstandard subnet must include this modifier. Specify this modifier after any optional modifiers. Do not specify this modifier if you specify a CIDR bitmask (*/bitmask*). Do not specify this modifier with the `change` command.

-nofragtopmtu

Specifies that IP datagram fragmentation is disabled for this route.

-nopmtudisc

Specifies that path MTU discovery is disabled for this route.

-olddev *device*

Specifies the old interface device that you want to change in the routing entry. Use this modifier with the `change` command only to move a route from one interface to another.

-oldgateway *name*

Specifies the old gateway that you want to change in the routing entry. Use this modifier with the `change` command only.

-precedence *value*

Sets the precedence of the route to the value specified by *value*. Among equivalent routes to the same destination, the route with the lower precedence is preferred.

-recvpipe *bandwidth*

Sets this route's inbound delay bandwidth product (in bytes) to the value specified by *bandwidth*.

-reject

Specifies that this route is a reject route. Packets sent to reject routes are dropped, and messages designating the route as unreachable are sent to the packet originators. This is different from a normal route, which allows packets to be forwarded out on it. You must specify `127.0.0.1` (localhost) as the gateway argument.

-rtt *time*

Sets this route's round-trip time (in microseconds) to the value specified by *time*.

-rttvar *variance*

Sets this route's round-trip time variance (in microseconds) to the value specified by *variance*.

-sendpipe *bandwidth*

Sets this route's outbound delay bandwidth product (in bytes) to the value specified by *bandwidth*.

-ssthresh *threshold*

Sets this route's outbound gateway buffer limit (in bytes) to the value specified by *threshold*.

Examples

The following examples show how to use the `route` utility.

1. `TCPIP> route add default 128.32.0.130`

   The example shows how to add gateway 128.32.0.130 as a default gateway.

2. `TCPIP> route add -host milan 128.32.0.130`

   The example shows how to add a route to host `milan` via gateway `128.32.0.130`.

3. `TCPIP> route delete -host milan 128.32.0.130`

   The example shows how to delete an existing route via gateway `128.32.0.130` to host `milan`.

4. `TCPIP> route add -precedence 1 -host milan 128.32.0.130`

   The example shows how to add a route with a precedence value of 1 to host `milan` via gateway `128.32.0.130`.

5. `TCPIP> route change -oldgateway 128.32.0.130 -oldinterface le0 -_TCPIP> -host milan 128.32.10.101`

   The example shows how to change an existing route for host `milan` via gateway `128.32.0.130` to use a new gateway `128.32.10.101`.

sysconfig

sysconfig — Maintains the kernel subsystem configuration.

Syntax

`sysconfig - -c | -d | -m | -q | "-Q" | -r | -s | -u [`*`subsystem-name`*`] [`*`attribute-list`*`]`

Description

The `sysconfig` command queries and modifies the in-memory subsystem configuration. Use this command to add subsystems, reconfigure subsystems that are already in memory, query subsystems, and unconfigure and remove subsystems.

The `sysconfig` utility allows you to modify the value of subsystem attributes, as long as the subsystem supports run-time modifications.

When you configure a subsystem using the `-c` flag, you make that subsystem available for use. If the subsystem is loadable, the `sysconfig` command loads the subsystem and then initializes the value of its attributes.

To modify the value of a subsystem attribute, use the `-r` (reconfigure) flag. Specify the subsystem attributes and values on the command line. The `sysconfig` utility modifies the named attributes by storing the value you specify in them. The modifications take effect immediately.

To get information about subsystem attributes, use either the `-q` flag or the `"-Q"` flag. You can specify an attribute list with both these flags. When you use the `-q` flag, the `sysconfig` command displays the value of attributes from the in-memory system configuration table.

When you use the `"-Q"` flag, the `sysconfig` utility displays the following information about each attribute you specify in the attribute list or, if you omit the attribute list, every attribute for the specified subsystem. You must enclose the `"-Q"` flag in quotation marks to preserve its case.

- Attribute datatype.
- Operations supported by the attribute. The op code indicates whether the attribute can be:
  - Configured
  - Reconfigured
  - Queried

  For example, if `op=RCQ`, the attribute can be configured (using the `sysconfigdb` utility), reconfigured (using the `sysconfig -r` command), and queried (using the `sysconfig -q` command).
- Minimum and maximum allowed attribute values.

To get information about the state of subsystems, use the `-s` flag. This flag provides a list of the subsystems that are currently loaded and configured. If you specify *subsystem-name*, the command displays information about the state of that subsystem. Each subsystem can have one of three states:

- Loaded and configured (available for use)
- Loaded and unconfigured (not available for use but still loaded)

  This state applies only to static subsystems, which can be unconfigured but cannot be unloaded.
- Unloaded (not available for use)

  This state applies only to loadable subsystems, which are automatically unloaded when you unconfigure them with the `sysconfig -u` command.

Subsystems that are not being used can be unconfigured using the `-u` flag. Unconfiguring subsystems can free up kernel memory, making it available for other uses. You can unconfigure any static or loadable subsystem that supports run-time unconfiguration. If you unconfigure a loadable subsystem, that subsystem is also unloaded from the kernel.

You can use the `sysconfig` command to display the value of attributes on the local system. If you want to configure, reconfigure, or unconfigure a subsystem, you must be authorized to modify the kernel configuration. Only users who have a system group UIC or who have an account with SYSPRV, BYPASS, or OPER privilege can configure, reconfigure, or unconfigure the subsystems.

Parameters

subsystem-name

Specifies the subsystem on which you want to perform the operation. The *subsystem-name* argument is required for all flags except `-s` and `-m`. If you omit *subsystem-name* when you use the `-s` or `-m` flag, the sysconfig utility displays information about all loaded subsystems.

attribute-list

Specifies attribute names and, depending on the operation, attribute values.

- For reconfigure (`-r`) operations, the *attribute-list* argument has the following format:

  ```
  attribute1=value1 attribute2=value2...
  ```

  Do not include spaces between the attribute name, the equals sign (=), and the value.

- For query attribute (`-q`) operations, the *attribute-list* argument has the following format:

  ```
  attribute1 attribute2...
  ```

The *attribute-list* argument is required when you use the `-r` flag and is optional with the `-q` flag. Any attribute list specified with other flags is ignored by the `sysconfig` utility.

Flags

-c

Configures the specified subsystem by initializing its attribute values and, possibly, loading it into memory. Use this command whether you are configuring a newly installed subsystem or one that was removed using the `sysconfig -u` command option.

-d

Displays the attribute settings in the SYSCONFIGTAB.DAT file for the specified subsystem.

-m

Queries the mode for the specified subsystems. A subsystem's mode can be static or dynamic. If you omit the subsystem name, `sysconfig` displays the mode of all the configured subsystems.

-q

Queries attribute values for the configured subsystem specified by *subsystem-name*. If you omit the attribute list, values for all the specified subsystem's attributes are displayed.

"-Q"

Queries information about attributes of the configured subsystem specified by `subsystem-name`. The information includes the attribute data type, the operations supported, and the minimum and maximum values allowed for the attribute. Note that the minimum and maximum values refer to length and size for attributes of `char` and `binary` types, respectively. If you omit the `attribute-list` argument, information about all attributes in the specified subsystem is displayed.

-r

Reconfigures the specified subsystem. You must supply the subsystem name and the attribute list when you use this flag.

-s

Queries the subsystem state for the specified subsystems. If you omit the subsystem name, `sysconfig` displays the state of all the configured subsystems.

-u

Unconfigures and, if the subsystem is loadable, unloads the specified subsystem from the kernel.

Examples

The following examples show how to use the `sysconfig` command.

1. ```
   TCPIP> sysconfig -s
   inet: loaded and configured
   net: loaded and configured
   socket: loaded and configured
   iptunnel: loaded and configured
   ipv6: loaded and configured
   snmpinfo: loaded and configured
   ```

   This example shows how to display the subsystems and their status.

2. ```
   TCPIP> sysconfig -q net
   net:
   ifnet_debug = 0
   ifqmaxlen = 1024
   lo_devs = 1
   lo_def_ip_mtu = 4096
   nslip = 0
   ```

   This example shows how to display subsystem attributes and their values.

3. ```
   TCPIP> sysconfig -s net
   net: loaded and configured
   ```

   This example shows how to query the state of a particular subsystem.

sysconfigdb

sysconfigdb — Manages the subsystem configuration database.

Syntax

`sysconfigdb {-a | -u} [-t` *`target`*`] -f` *`file subsystem-name`*

`sysconfigdb {-m | -r} [-t` *`target`*`] -f` *`file`* `[subsystem-name]`

`sysconfigdb -d [-t` *`target`*`]` *`subsystem-name`*

`sysconfigdb -l [-t` *`target`*`] [`*`subsystem-name`*`,...]`

Description

The `sysconfigdb` utility is used to manage the subsystem configuration table (TCPIP$ETC:SYSCONFIGTAB.DAT). To specify another file as a target file, use the `-t` flag.

To modify a target file, create a stanza file. This stanza file contains the name of one or more subsystems, each with a list of attributes and their values, as described in Section 2.1.3.1.
   ```

Modifications that you make to the `sysconfigtab` are changed the next time the subsystem is reloaded.

When the target file is another file, there is no synchronization with the subsystem configuration database.

Restrictions

You must have system management privileges to run the sysconfigdb utility to modify the system configuration table.

Parameters

subsystem-name

Specifies a subsystem that contains the attributes you want to modify. The subsystem name and attributes are in a stanza input file.

You must specify the subsystem name when deleting (`-d`), adding (`-a`), or replacing (`-u`) a subsystem.

In other cases, when you do not specify a subsystem name, the operation is attempted for all the subsystems and attributes specified in the input file.

Flags

-a

Adds the specified subsystem entry to the target file.

-d

Deletes the specified subsystem entry from the target file.

-f *file*

Specifies the input file, a stanza file that contains entries for one or more subsystems. The default target file is the SYSCONFIGTAB.DAT file. Specify another target file by using the `-t` target flag.

-l

Lists the specified subsystem entries in the target file. If you do not specify a subsystem name, all subsystem entries in the target file are listed. The SYSCONFIGTAB.DAT file is the default target file.

-m

Merges subsystem attributes specified in the input file with the subsystem attributes in the target file. If you do not specify a subsystem name, all subsystem entries in the input file are merged. The SYSCONFIGTAB.DAT file is the default target file.

-r

Removes the subsystem entries specified in the input file from the target file. The only entries removed are those that have attribute names and values that exactly match those in the input file.

If you do not specify the subsystem name, all subsystem entries in the input file with attributes that match are removed from the target file. The SYSCONFIGTAB.DAT file is the default target database file.

-t *file*

Specifies the target file for the operation. If you do not specify this flag, the default target file is the SYSCONFIGTAB.DAT file.

-u

Replaces a subsystem entry in the target file with the subsystem entry specified in the input file.

Examples

The following examples show how use the `sysconfigdb` utility.

1. ```
   $ TCPIP
   TCPIP> sysconfigdb -u -f table_mgr.stanza table_mgr_1
   ```

   This command replaces the `table_mgr_1` entry in the SYSCONFIGTAB.DAT file with the information in the TABLE_MGR.STANZA file for the `table_mgr_1` subsystem. The command updates the in-memory copy of the subsystem configuration database to match the modified SYSCONFIGTAB.DAT file.

2. ```
   TCPIP> sysconfigdb -m -f table_mgr.stanza tbl_mgr_2
   ```

   This command merges the `tbl_mgr_2` information from the `table_mgr.stanza` file with the information already in the `tbl_mgr_2` entry in the SYSCONFIGTAB.DAT file. The command updates the in-memory copy of the subsystem configuration database to match the modified SYSCONFIGTAB.DAT file.

3. ```
   TCPIP> sysconfigdb -l table_mgr_1
   table_mgr_1:
                 size = 10
                 name = Ten-Element-Table
   ```

   This command lists the entry for the subsystem `table_mgr_1`. This command does not update the in-memory copy of the subsystem configuration database.

4. ```
   TCPIP> sysconfigdb -d table_mgr_1
   ```

   This command deletes the `table_mgr_1` entry from the SYSCONFIGTAB.DAT file and updates the in-memory copy of the subsystem configuration database to match the modified SYSCONFIGTAB.DAT file.

tcpdump

tcpdump — Provides dump analysis and packet capturing.

Syntax

```
tcpdump ["-B" | d | e | f | l | m | n | "-N" | "-O" | q | s | "-S" |
t | v | x | "-X"]
```

```
[-b buffers]

[-c count]

[-F file]

[-r file]

[-s snaplen]

[-w file] expression
```

Description

The `tcpdump` utility displays the headers and contents of packets on the network that match a boolean expression (filter). If no filter is supplied all packets processed by `tcpdump` will be displayed. The the packets that are processed can also be written to a binary file for later examination and filtering.

Parameters

expression

A boolean expression that provides a filter to select the packets to dump. If you do not specify the expression, all packets on the network are dumped. Otherwise, only packets that match the expression are dumped.

For information about specifying expressions, refer to Section 1.2.5.2.

Options

-b

Specifies the number of buffers used to communicate with the TCP/IP kernel. The default is 400 on Alpha systems and 50 on VAX systems.

"-B"

Displays buffer diagnostics showing when dropped packets occur. Use quotation marks to preserve the case of uppercase options.

-c

Exits after receiving count packets.

-d

Dumps the compiled packet-matching code to standard output and stops.

-e

Displays the link-level header on each dump line.

-f

Displays foreign internet addresses numerically rather than symbolically.

"-F" ***file***

Uses *file* as input for the filter expression. Any additional expressions on the command line are ignored. Use quotation marks to preserve the case of uppercase options.

-l

Buffers the `stdout` line. This is useful if you want to see the data while capturing it.

-m

Enables multiline output from some protocols. This affects most ONC RPC decoding, as those protocols are often difficult to display on a single line.

-n

Does not convert addresses (for example, host addresses and port numbers) to names.

"-N"

Does not display domain name qualification of host names. For example, with this option, `tcpdump` displays `nic` instead of `nic.ddn.mil`. Use quotation marks to preserve the case of uppercase options.

"-O"

Does not run the packet-matching code optimizer. This is useful only if you suspect a bug in the optimizer. Use quotation marks to preserve the case of uppercase options.

-q

Quick (quiet) output. Displays less protocol information so output line are shorter.

-r ***file***

Reads packets from *file* (which was created with the `-w` option). Standard input is used if a hyphen (-) is used to specify the file.

-s ***snaplen***

Displays the number of bytes of data from each packet as specified by the value of *snaplen*, rather than the default of 68. The default of 68 bytes is adequate for IP, ICMP, TCP, and UDP, but may truncate protocol information from name server and NFS packets. Packets truncated because of a limited snapshot are indicated in the output with `[|`*`proto`*`]`, where *proto* is the name of the protocol level at which the truncation has occurred.

Note

Taking larger snapshots both increases the amount of time it takes to process packets and decreases the amount of packet buffering. This may cause packets to be lost. You should limit the value of `snaplen` to the smallest number that will capture the protocol information you need.

"-S"

Displays absolute, rather than relative, TCP sequence numbers. Use quotation marks to preserve the case of uppercase options.

-t

Does not display a timestamp on each dump line.

-tt

Displays an unformatted timestamp on each dump line.

-v

Displays verbose output. For example, the time to live and type of service information in an IP packet is displayed. If `-m` is also specified, ONC RPC packets sent using TCP are decoded twice: first as RPC, then as TCP. By default, the TCP decoding is suppressed.

-vv

Displays detailed verbose output. For example, additional fields are displayed from NFS reply packets.

-w *file*

Writes the raw packets to file rather than parsing and displaying them. They can later be displayed with the `-r` option. Standard output is used if a hyphen (-) is used to specify the file.

-x

Displays each packet (minus its link level header) in hexadecimal format.

The smaller of the entire packet or *snaplen* bytes is displayed.

"-X"

Displays packets in both hexadecimal and ASCII formats. Use quotation marks to preserve the case of uppercase options.

Examples

1. `$ tcpdump host sundown`

   This example shows how to use the tcpdump utility to display all packets arriving at or departing from host `sundown`.

2. `$ tcpdump host sundown and ( hot or ace )`

   This example shows how to use the tcpdump utility to display traffic between sundown and either host hot or host ace.

3. `$ tcpdump ip host ace and not helios`

   This example shows how to use the tcpdump utility to display all IP packets between ace and any host except `helios`.

4. `$ tcpdump net office`

   This example shows how to use the tcpdump utility to display all traffic between local hosts and hosts on the network `office`.

5. `$ tcpdump gateway snup and (port 21 or 20)`

   This example shows how to use the tcpdump utility to display all FTP traffic through Internet gateway `snup`.

6. `$ tcpdump ip and not net localnet`

   This example shows how to use the tcpdump utility to display traffic neither sourced from nor destined for local hosts. If your network is connected to one other network by a gateway, this command does not produce any results on your local network.

7. `$ tcpdump tcp[13] & 3 != 0 and not src and dst net localnet`

   This example shows how to use the tcpdump utility to display the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a nonlocal host.

8. `$ tcpdump gateway snup and ip[2:2] > 576`

   This example shows how to use the tcpdump utility to display IP packets longer than 576 bytes sent through gateway snup.

9. `$ tcpdump ether[0] & 1 = 0 and ip[16] >= 224`

   This example shows how to use the tcpdump utility to display IP broadcast or multicast packets that were not sent using Ethernet broadcast or multicast.

10. `$ tcpdump icmp[0] != 8 and icmp[0] != 0`

    This example shows how to use the tcpdump utility to display all ICMP packets that are not echo requests or replies (that is, not PING packets).

11. `$ tcpdump -s 1500 -envv ipv6 and udp port 521`

    This example shows how to use the `tcpdump` utility to display all RIPv6 packets.

12. `$ tcpdump -s 1500 -envv ipv6 and ether host a:b:c:d:e:f`

    This example shows how to use the tcpdump utility to display all IPv6 packets arriving at or departing from a host with the Ethernet address `a:b:c:d:e:f`

TCPTRACE

TCPTRACE — Traces packets between two hosts.

Syntax

`TCPTRACE` *`host`* `[/BUFFERS=`*`n`*` | /FULL | /OUTPUT=`*`file`*` | /PACKETS=`*`n`*` | /PORT=`*`option`*` | /PROTOCOL=`*`option`*`]`

Description

TCPTRACE traces packets as they travel between the local and remote host. You can trace all packets or you can use command qualifiers to monitor only those packets of interest.

Qualifiers

/BUFFERS=*n*

Optional. The default is 100. Specifies the number of buffers that TCPTRACE allocates for temporary storage. These buffers must be locked into the working set, so the number can be:

- Decreased (to be accommodated in the working set)
- Raised (to prevent the dropping of trace packets)

/FULL

Optional. The default is brief display.

Displays the packet's contents.

/OUTPUT=*file*

Optional. The default is screen display. Redirects the output from screen to the specified file. If this file name already exists, the output is appended to it.

/PACKETS=*n*

Optional. The default it 10.

Stops the trace after the specified number of packets is displayed.

/PORT={LOCAL=*n* | REMOTE=*n*}

Optional for port number. The default is that all traffic is displayed.

Required for port type. Filters the trace to the specified port.

/PROTOCOL={ARP | ICMP | IP | TCP | UDP}

Optional. The default is /PROTOCOL=IP.

Filters the specified protocol.

Examples

The following examples show how to use the TCPTRACE command.

1. `$ TCPTRACE HOST1 /FULL /PORT=(REMOTE=21)`

   This example shows how to use the `TCPTRACE` command to trace packets between the local system and `Host1`. `TCPTRACE` filters all packets except those packets directed to port 21 on the remote host.

2. ```
   $ TCPTRACE HOST2 /PORT=(LOCAL=23, REMOTE=1056) -
   _$ /FULL /PACKETS=30 /OUTPUT=TELNET_TRACE.TXT
   ```

   This example shows how to use the `TCPTRACE` command to trace packets between the local system and `Host2`. `TCPTRACE` filters all packets except those packets directed to port 23 on the
   ```

local host and port 1056 on the remote host. The trace results are output to a file with the name TELNET_TRACE.TXT. The trace stops after 30 packets meeting the specifications are encountered.

traceroute

traceroute — Displays the route that packets take to the network host.

Syntax

`traceroute [-m max_ttl] [-n] [-p port] [-q nqueries] [-r] [-s src_addr] [-v] [-w waittime] host [packetsize]`

Description

The Internet is a large and complex aggregation of network hardware connected together by gateways. The `traceroute` command tracks the route that packets follow from gateway to gateway. The command uses the IP protocol time-to-live (TTL) field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to a particular host.

The only mandatory parameter is the destination host name or IP address. The default probe datagram length is 38 bytes, but you can increase this value by specifying a packet size (in bytes) after the destination host name. This is useful when the `-f` option is given for MTU discovery along the route. You should start with the maximum packet size for your own network interface (if the given value is even bigger, `traceroute` attempts to select a more appropriate value). If no packet size is given when using the `-f` option, `traceroute` determines the initial MTU automatically. To track the route of an IP packet, `traceroute` launches UDP probe packets with a small TTL (time-to-live) and then listens for an ICMP “time exceeded” reply from a gateway. Probes start with a TTL of 1 and increment by one until either an ICMP “port unreachable” is returned (indicating that the packet reached the host) or until the maximum number of hops is exceeded (the default is 30 hops and can be changed with the `-m` option). At each TTL setting, three probes are launched (the number can be changed with the `-q` option), and `traceroute` displays a line showing the TTL, address of the gateway, and round-trip time of each probe. If the probe answers come from different gateways, `traceroute` displays the address of each responding system. If there is no response within a 3-second timeout interval (which can be changed with the `-w` option), `traceroute` displays an asterisk (*) for that probe.

To prevent the destination host from processing the UDP probe packets, the destination port is set to an unlikely value. If necessary, you can change the destination port valued with the `-p` option.

Note

This program is intended for use in network testing, measurement, and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, do not use `traceroute` during normal operations or from automated scripts.

Flags

-A

Looks up the AS-number (autonomous system) for each hop's network address at the `whois` server specified by the `-h` option.

-a

If the destination host has multiple addresses, `traceroute` probes all addresses if this option is set. Normally, only the first address as returned by the resolver is attempted.

-c stoptime

Specifies a delay (in seconds) to pause between probe packets. This can be necessary if the final destination is a router that does not accept undeliverable packets in bursts.

-f

Disables IP fragmentation. If the given packet size is too big to be handled unfragmented by a machine along the route, a “fragmentation needed” status is returned, and the indicator !F is printed. If a gateway returns the proper MTU size to be used, `traceroute` automatically decreases the packet size to this new value. If the proper MTU size is not returned, `traceroute` chooses a smaller packet size.

-g gateway

Enables the IP LSRR (loose source record route) option. This is useful for asking how somebody at the specified gateway reaches a particular target.

-h server

Specifies the name or IP address of the `whois` server that is contacted for the AS-number lookup, if the `-A` option is given.

-i initial_ttl

Sets the starting time-to-live value to `initial_ttl`, to override the default value of 1. Effectively this skips processing for intermediate hosts that are less than `initial_ttl` hops away.

-k

Keeps the connection to the `whois` server permanently open. This speeds lookups considerably, because a connection setup for each individual lookup is not necessary. However, not all `whois` servers support this feature.

-l

Prints the value of the TTL field in each packet received. (This flag can be used to help detect asymmetric routing.)

-m *max_ttl*

Sets the maximum time-to-live (maximum number of hops) used in outgoing probe packets. The default is 30 hops, which is the same default used for TCP connections.

-N

Displays the network name for each hop. If a BIND resolver cannot be reached, network names are retrieved just from the `/etc/networks` file.

-n

Displays the hop IP addresses using dotted-decimal notation. This saves a name server address-to-name lookup for each gateway found on the path. It also prevents a reverse lookup for numeric

dotted-quad addresses given on the command line, such as destination `host` or `-g gateway` addresses.

-p port

Sets the base UDP port number used in probes. (The default value is 33434.) The `traceroute` command presumes that nothing is listening on UDP ports `base` to `base+nhops-1` at the destination host (so an ICMP "port unreachable" message is returned to terminate the route tracing). If another process is listening on a port in the default range, use this option to pick an unused port range.

-Q maxquit

Stops probing this hop after the number of consecutive timeouts specified by `maxquit` are detected. The default value is 5. Useful in combination with `-S` if you have specified a big `nqueries` probe count.

-q *nqueries*

Sets the number of probes launched at each TTL setting. The default is 3.

-r

Bypasses the normal routing tables and sends directly to a host on an attached network. If the host is not on a directly attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (for example, after the interface was dropped by ROUTED or GATED).

-S

Prints a summary of per-hop minimum/average/maximum rtt (round-trip time) statistics. This flag suppresses the per-probe rtt and TTL reporting. To obtain more detailed statistics, increase the default `nqueries` probe count. For more information, see also the `e` option.

-s *src_addr*

Uses the following IP address (which must be given as an IP number, not as a host name) as the source address in outgoing probe packets. On hosts with more than one IP address, this option can be used to force the source address to be something other than the IP address of the interface on which the probe packet is sent. If the IP address is not one of this machine's interface addresses, an error is returned and nothing is sent.

-v

Lists any received ICMP packets other than TIME_EXCEEDED and UNREACHABLE.

-w *waittime*

Sets the time (in seconds) to wait for a response to a probe. The default is 3 seconds.

Parameters

host

Specifies the name or IP address of the destination host. This parameter is required.

packetsize

Specifies the default length for a probe datagram. This parameter is optional. The default is 38 bytes.

Examples

The following examples show how to use the `traceroute` command.

1.
```
localhost> traceroute nis.nsf.net

traceroute to nis.nsf.net (35.1.1.48), 30 hops max, 56 byte packet

 1  helios.ee.lbl.gov (128.3.112.1)  19 ms  19 ms  0 ms
 2  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  39 ms 19 ms
 3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  39 ms 19 ms
 4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  39 ms  40 ms  39 ms
 5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  39 ms  39 ms  39 ms
 6  128.32.197.4 (128.32.197.4)  40 ms  59 ms  59 ms
 7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  59 ms
 8  129.140.70.13 (129.140.70.13)  99 ms  99 ms  80 ms
 9  129.140.71.6 (129.140.71.6)  139 ms  239 ms  319 ms
10  129.140.81.7 (129.140.81.7)  220 ms  199 ms  199 ms
11  nic.merit.edu (35.1.1.48)  239 ms  239 ms  239 ms
```

   This `traceroute` command displays the route that packets take to a remote host. In this example, note that display lines 2 and 3 are identical. This is due to a bug in the kernel on the second hop system, `lbl-csam.arpa`, that forwards packets with a zero TTL. (This is a bug in the distributed version of BSD Version 4.3.)

2.
```
localhost> traceroute allspice.lcs.mit.edu

traceroute to allspice.lcs.mit.edu (18.26.0.115), 30 hops max

 1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
 2  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  19 ms 19 ms
 3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms 19 ms
 4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  19 ms  39 ms  39 ms
 5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  20 ms  39 ms  39 ms
 6  128.32.197.4 (128.32.197.4)  59 ms  119 ms  39 ms
 7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  39 ms
 8  129.140.70.13 (129.140.70.13)  80 ms  79 ms  99 ms
 9  129.140.71.6 (129.140.71.6)  139 ms  139 ms  159 ms
10  129.140.81.7 (129.140.81.7)  199 ms  180 ms  300 ms
11  129.140.72.17 (129.140.72.17)  300 ms  239 ms  239 ms
12  * * *
13  128.121.54.72 (128.121.54.72)  259 ms  499 ms  279 ms
14  * * *
15  * * *
16  * * *
17  * * *
18  ALLSPICE.LCS.MIT.EDU (18.26.0.115)  339 ms  279 ms 279 ms
```

   In this example, gateways 12, 14, 15, 16, and 17 either do not send ICMP “time exceeded” messages or send them with a TTL too small to reach the local host. Further investigation is required to determine the cause. For example, by contacting the system administrators for gateways 14 through 17, you could discover that these gateways are running the MIT C Gateway code that does not send “time exceeded” messages.