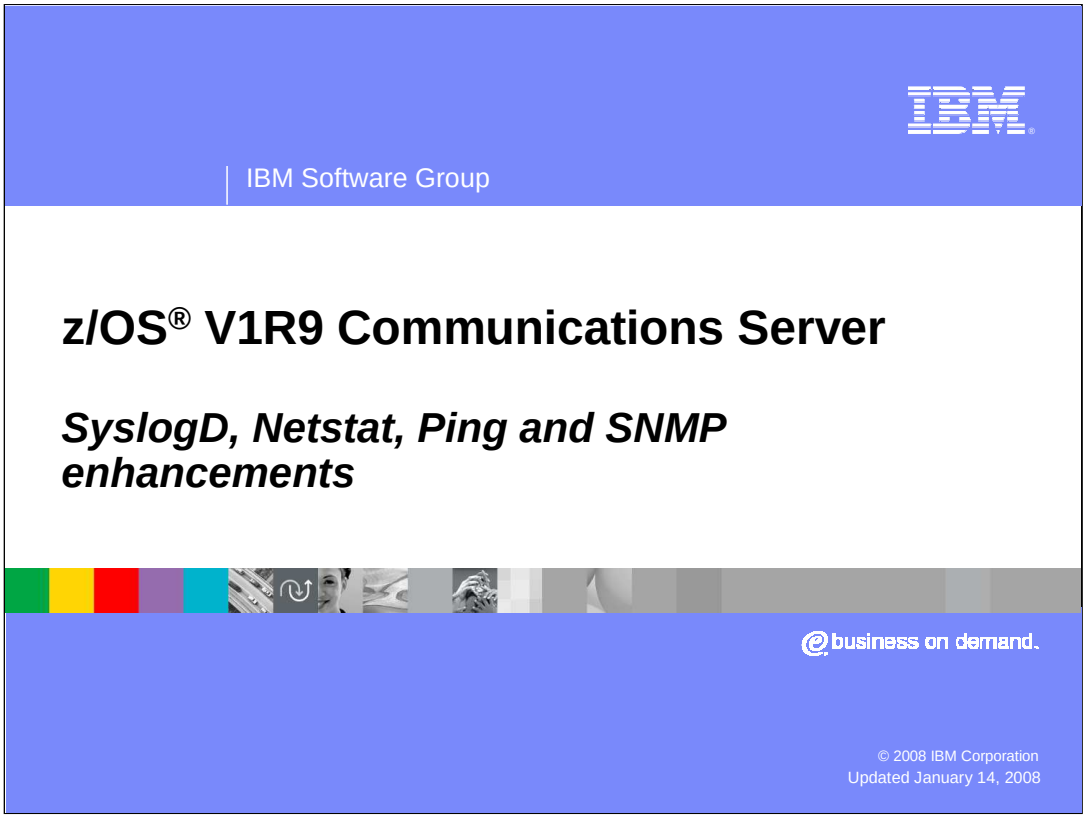


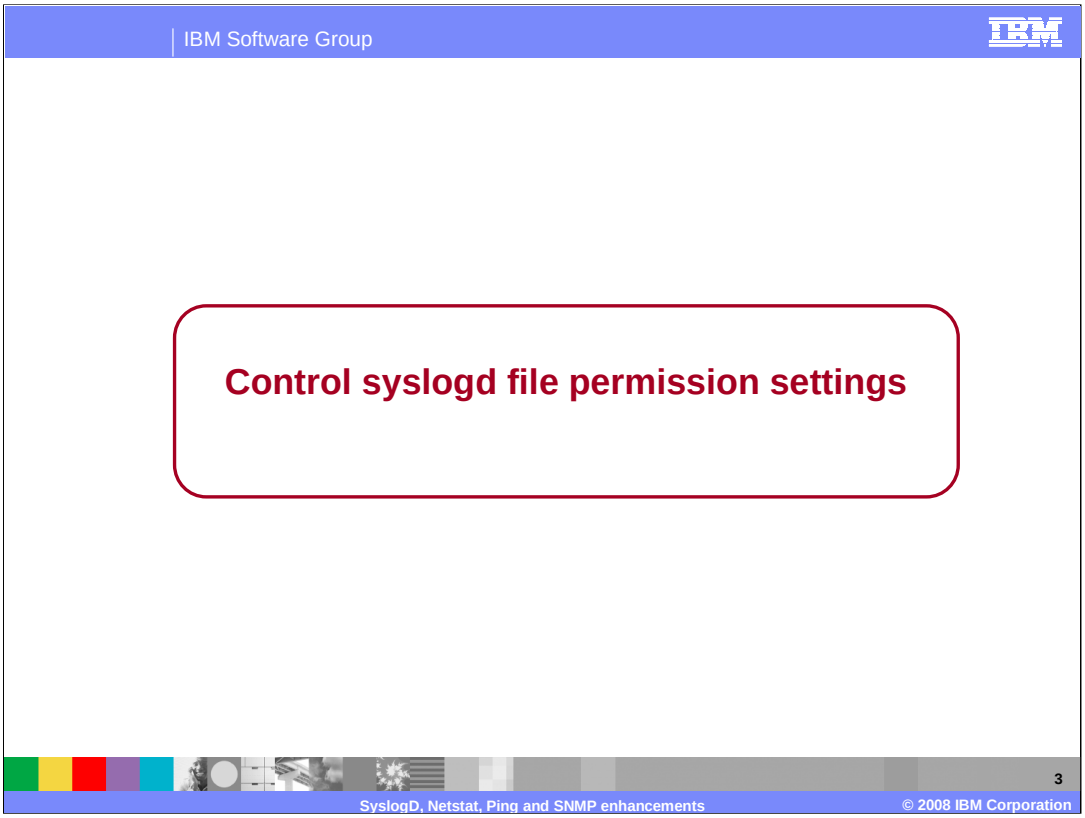
This presentation describes the changes made in z/OS V1R9 Communications Server in the area of systems management to control syslogd file permission settings, enhance NETSTAT ALL/-A report to indicate sockets storage use, enhance the Ping command to detect the network MTU, and provide a programming interface for SNMP.

Agenda

- Control syslogd file permission settings
- Enhance NETSTAT ALL/-A report to indicate sockets storage use
- Ping command detection of network MTU
- Provide a Programming interface for SNMP

z/OS V1R9 Communications Server has several enhancements in the area of systems management. All of these will be discussed in detail.

This presentation will also cover the new function to allow applications to store identifying data on TCP connections.

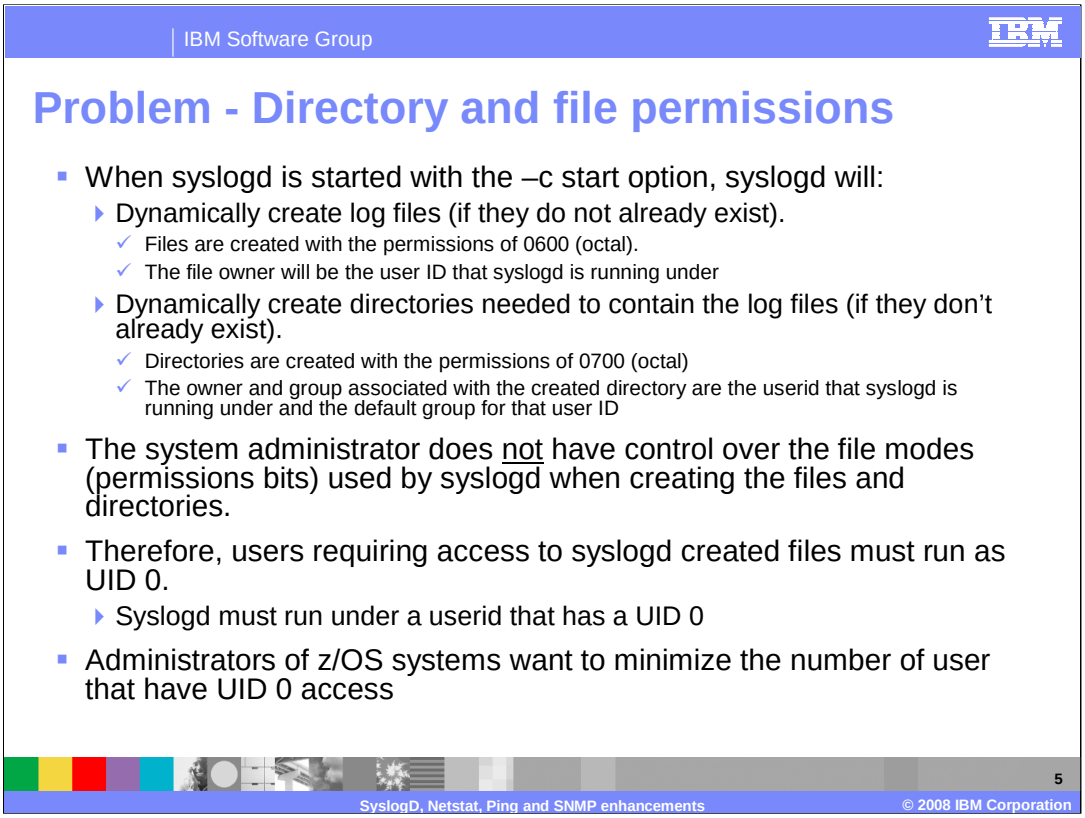


This section describes the changes to Control syslogd file permission settings.

Background information

- syslogd - the UNIX® syslog daemon
  - Accepts log messages from local applications or remote syslog daemons (if so configured).
  - Uses a configuration file containing rules to determine log message destinations. Destinations may include:
    - ✓ A UNIX file system file
    - ✓ The MVS system console
    - ✓ Logged on UNIX System Services users
    - ✓ SMF
    - ✓ OPERLOG
    - ✓ Remote syslog daemons (using UDP)

syslogd is the UNIX syslog daemon. It is a program that runs on z/OS and accepts log messages generated by local applications or from remote syslog daemons running on other systems in the network. syslogd uses a default configuration file named /etc/syslog.conf or you may start it by specifying a configuration file as a start parameter. The syslogd configuration file contains rules that tell syslogd where to log messages based on the message's facility and priority. The z/OS syslogd supports the following destinations for messages: A UNIX file system file, the MVS system console, logged on users, SMF, OPERLOG and forwarding using UDP to remote syslog daemons elsewhere in the network.



A common destination for syslogd messages is a local UNIX file system file. By default, syslogd will not create log files if they do not exist. However, if the –c option is used when starting syslogd, syslogd will create the destination file if it does not exist. syslogd will also create directories if needed to contain the file. The ability to create files dynamically allows for improved organization since log files can be named using current dates or days of the week. However, the ownership and access permissions for the files and directories created by syslogd has always been fixed and not user-modifiable. The owner of files and directories created by syslogd is always UID 0 since syslogd must run under a UID 0 userid. The default permissions used by syslogd when creating files or directories require that any user that needs access to the files must also be UID 0. Administrators of z/OS systems want to minimize the number of users that have UID 0 access.

Syslogd creates files with permissions bits of 0600 (octal). This octal value indicates that the owner of the file has: read access and write access. Syslogd creates directories with the permissions bits of 0700 (octal). This octal value indicates that the owner of the directory has: read access, write access and traverse access. See the description of the chmod command in the UNIX System Services Command Reference book for more information on permissions for directories and files. The syslog daemon must always be run by a userid that has UID 0.

Solution - Directory and file permissions

- Allow system programmer to specify the default permissions to be used by syslogd when creating directories and files with two new start options
  - -D octal_value default permissions for directories
  - -F octal_value default permissions for files
- Allow the default permissions to be overridden on a rule-by-rule basis if needed. Two new configuration options on rules with file destinations
  - -D octal_value permission to use for directory
  - -F octal_value permission to use for file
  - Override the corresponding values specified when starting syslogd
- -D and –F are only valid when the destination is a regular UNIX file system file
- You can specify –F and –D independently, together or not at all

With this solution, UID 0 user IDs are no longer required in order to access syslogd log files. With z/OS V1R9, syslogd can be started with two new start options. The –D option allows the system programmer to specify the default permissions that should be used by syslogd when creating a directory. The –F option allows the system programmer to specify the default permissions that should be used by syslogd when creating a file. These are optional and one or both can be used.

Additionally, the default directory and file permissions can be overridden on a rule-by-rule basis. The –D option is used to specify the permission bits to use for creating a directory for the rule. The –F option is used to specify the permission bits to be used when creating a file for the rule. These are optional and one or both may be used. If used, the values override the corresponding start option values for –D and –F. These options are only valid when the destination is a regular USS file system file. You may specify –D or –F independently or together. If a value is not overridden on a rule with –D or –F, the value from the corresponding start option is used. If the corresponding start option was not specified, then the default value is used. The default permissions for directories is 0700 octal and the default permissions value for files is 0600 octal. Note that the new options are case-sensitive and must be entered in uppercase.

As always, in order for syslogd to create any directories or files, the –c start option must be used.

New messages

- FSUM1245 Incorrect value for %s parameter

Example:
FSUM1245 Incorrect value for -D parameter

- FSUM1246 %s must be specified with %s

Example:
FSUM1246 -c must be specified with -D

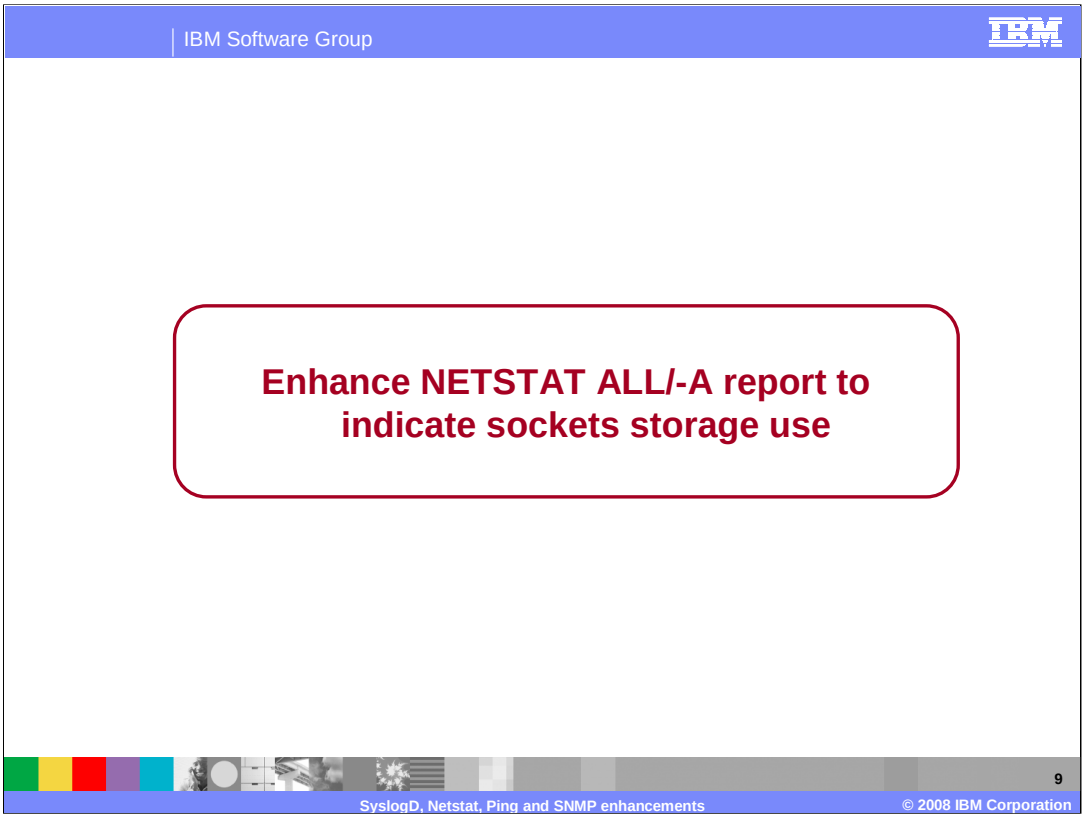
This slide lists the two new messages in syslogd for z/OS V1R9.
The possible values for –D that can be OR'ed together are
1000 sticky bit (restricted delete)
0400 owner read
0200 owner write
0100 owner list directory
0040 group read
0020 group write
0010 group list directory
0004 other read
0002 other write
0001 other list directory
Only bits listed here are allowed.

The possible values for –F that can be OR'ed together are:
0400 owner read
0200 owner write
0040 group read
0020 group write
0004 other read
0002 other write
Only bits listed here are allowed.

Things to think about

- There are no migration concerns. You must change the way you start syslogd or change the syslogd configuration file for this new function to have any effect.
- You must use the –c (dynamic file create) start option in order to use the new –D or –F start or configuration options.
- Directories and files that already exist are not affected by this new function.
- Before using –F or –D you must carefully consider and understand what permissions values are safe and appropriate for your systems.

There are no migration concerns with the new –D and –F start options. If you do not use the new options, syslogd will operate the same as it did in prior releases. You must use the –c option in order to use the new –D and –F start options or the new –D and –F configuration options. Directories and files that already exist are not affected by the –D or –F start options or configuration options. Before changing the default values for directory and file permissions you must be sure you understand what values are safe and appropriate for your particular systems.



This section describes the enhancements made to the Netstat ALL/-A commands to assist in determining a sockets storage usage.

Problem statement - No easy way to detect problem applications

- Data waiting to be read by an application or to be sent for an application is stored in CSM storage (ECSA/Dataspace) and referenced by TCPIP control blocks in ECSA
- An application (local or remote) may have a problem reading its data causing CSM and ECSA storage growth
- There is no easy way to determine which connection or application is causing storage growth.

For each TCP connection TCPIP maintains a send and receive buffer. The size used is specified on the TCPCONFIG parameters TCPRCVBufrSize and TCPSENDBfrsize or overridden by the application by using the SO_SNDBUF and SO_RCVBUF options in a SetSockopt command.

The size of the receive buffer is passed to the remote side of the connection during connection establishment as the window size. The window size is used to control how much data the remote side can send and varies during the connection as data accumulates on the receive queue waiting for the local application to issue read requests.

The size of the send buffer is used to control the amount of data being sent by the application to the remote host. If TCPIP is unable to immediately send the data when asked by the application (typically because the remote side has lowered their window size) then TCPIP will hold the data on the send queue. Once the queue reaches the send buffer size then TCPIP will not honor any additional send requests from the application on that connection until TCPIP is able to send some of the data already waiting on the send queue.

For each UDP socket TCPIP does not queue send data but will queue up received data waiting for the application to read it. The only limit on how much received data can be queued is by specifying UDPQUEUELIMIT on the UDPCONFIG statement, otherwise there is no limit on how much can be queued.

For both UDP sockets and TCP connections the message data is stored in CSM buffers. These may be in either ECSA or dataspace. Each message also has a control block structure that points to the message and contains information about the message and queue pointers. This control block is stored in ECSA.

TCPIP related ECSA and CSM storage growth may be attributed to application problems. A problem application may have stopped issuing reads for data for some reason or may be running too slowly to keep up with the speed that data is being received from it's remote partner. This can occur on either side of the connection and either can affect storage build up on z/OS TCPIP.

Note: the storage accumulating in these cases will be greater than just the message data on the send/receive queues since there are control blocks that are used to maintain these queues. Also, since multiple messages may be contained within a single CSM buffer, one held message can prevent the entire buffer from being released.

Currently there is not a way to easily tell if a CSM and ECSA storage growth is due to an application read problem and to identify which application.

Solution- Add additional information to Netstat

- Add additional information to the Netstat ALL/-A command output to help identify applications causing storage problems.
- For each TCP connection the amount of data on the receive queue (ReceiveDataQueued) and the send queue (SendDataQueued) will be displayed. If there is data on one of these queues then the date and time of the oldest message (OldQDate and OldQTime) on the queue will also be displayed.
- For each UDP socket the amount of data on the receive queue (ReceiveDataQueued) will be displayed with the number of messages (ReceiveMsgCnt). If there is data on the receive queue, the date and time of the oldest message (OldQDate and OldQTime) on the queue will also be displayed.

The Netstat ALL/-A command output provides detailed information about all connections. Additional messages have been added to the TCP connection and UDP socket information to specifically list receive and send queue data byte counts and the date and time of the oldest messages on these queues. For UDP sockets TCPIP maintains a count of the number of messages on the receive queue and this data is also now displayed.

Netstat command output (TCP)

```
Client Name: TCPCS                    Client Id: 0000000C
Local Socket: 9.67.115.5..23          Foreign Socket: 9.27.11.182..4665
  Last Touched:       16:46:15          State:              Establsh
  BytesIn:            0000001062        BytesOut:           0000000480
  SegmentsIn:         0000000019        SegmentsOut:        0000000019
  RcvNxt:             3296375906        SndNxt:             3296308452
  ClientRcvNxt:       3296375906        ClientSndNxt:       3296308452
  InitRcvSeqNum:      3296374843        InitSndSeqNum:      3296307971
  CongestionWindow:   0000340353        SlowStartThreshold: 0000016384
  IncomingWindowNum:  3296408638        OutgoingWindowNum:  3296341180
  SndWl1:             3296375906        SndWl2:             3296308452
  SndWnd:             0000032728        MaxSndWnd:          0000032768
  SndUna:             3296308452        rtt_seq:            3296308412
  MaximumSegmentSize: 0000065483        DSField:            00
  Round-trip information:
    Smooth trip time: 37.000            SmoothTripVariance: 101.000
  ReXmt:              0000000000        ReXmtCount:         0000000000
  DupACKs:            0000000000
  SockOpt:            00                TcpTimer:           00
  TcpSig:             00                TcpSel:             C0
  TcpDet:             F0                TcpPol:             00
  QOSPolicyRuleName:
  TTLSPolicy:         Yes
    TTLSRule:         TTLSRule1
    TTLSGrpAction:    TTLSGrpAction1
    TTLSEnvAction:    TTLSEnvAction1
    TTLSConnAction:   TTLSConnAction1 (Stale)
  ReceiveBufferSize:  0000016384        SendBufferSize:     0000016384
  ReceiveDataQueued:  000000002C
    OldQDate:         09/15/06          OldQTime:           03:36:32
  SendDataQueued:     000002C000
    OldQDate:         09/15/06          OldQTime:           03:36:32
```


This slide shows the output for a TCP connection. New messages/fields are shown in RED. If there is no data on a specific queue then the OldQDate and OldQTime message will not be displayed for that queue.

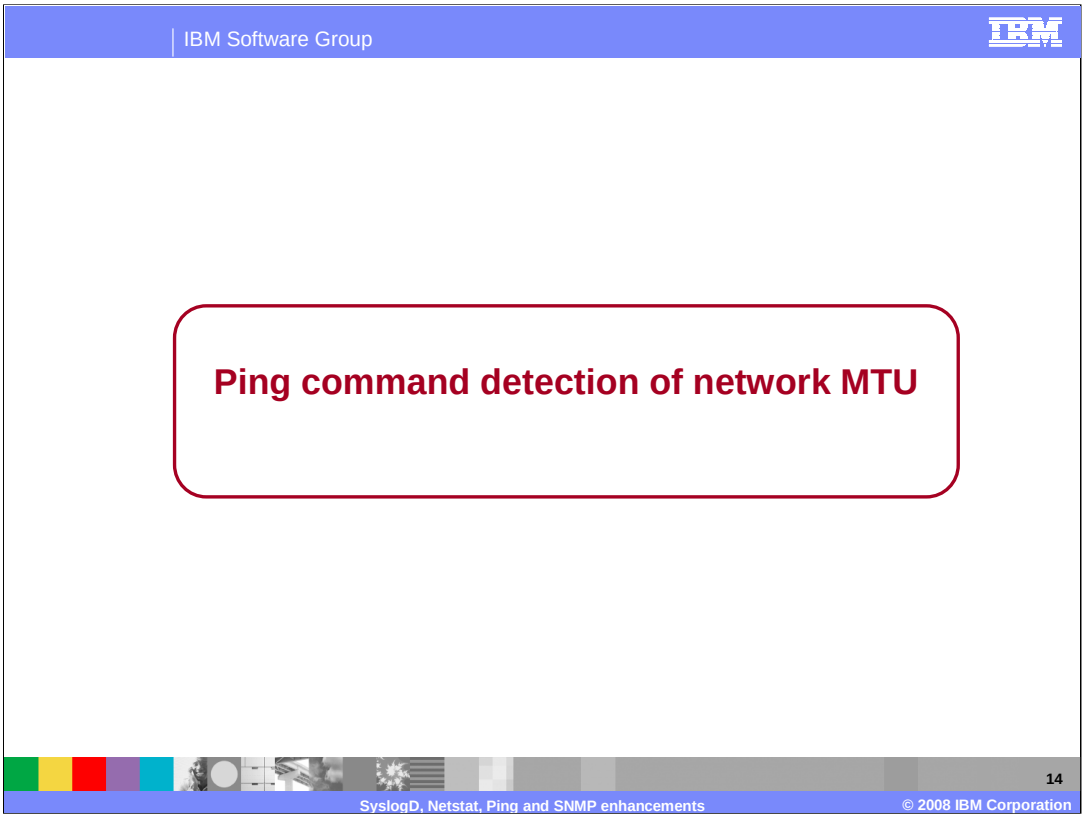
Netstat command output (UDP)

```
Client Name: APPV4                    Client Id: 00000015
  Local Socket: 0.0.0.0..2049
  Foreign Socket: 9.42.103.99..1234
    BytesIn:            00000000000000000200
    BytesOut:           00000000000000000100
    DgramIn:            00000000000000000010
    DgramOut:           00000000000000000005
    Last Touched:       16:00:29
    MaxSendLim:         0000065535        MaxRecvLim:         0000065535
    SockOpt:            00000000          DSField:            00
    QOSPolicyRuleName:
    ReceiveDataQueued:  0000005655        ReceiveMsgCnt:      0000000644
      OldQDate:         09/15/06          OldQTime:           03:36:32
```


This slide shows the output for a UDP socket.

New messages/fields are shown in RED.

If there is no data on a specific queue then the OldQDate and OldQTime message will not be displayed for that queue.



This section describes the changes to the Ping command to provide detection of network MTU size.

Background information

- **Ping command**
  - Used to determine if a host is active
  - Sends an IP packet, containing an ICMP or ICMPv6 echo request, to a destination host
  - Waits for an ICMP or ICMPv6 echo reply from the destination host
- **Maximum Transmission Unit (MTU)**
  - Largest size packet which can be sent on a network
- **Path MTU discovery**
  - Dynamic discovery of the largest MTU value which can be used to send packets to a destination host without causing fragmentation of the packets.
  - Enabled for IPv4 processing by specifying the PATHMTUDISCOVERY parameter on the IPCONFIG profile statement. Only triggered by TCP processing.
  - Always enabled for IPv6 processing and triggered for all processing
- **Fragmentation of IP Packets**
  - IPv4
    - ✓ Packets can be fragmented by any host
    - ✓ Setting the “don’t fragment” bit in the IP header prevents packet from being fragmented.
  - IPv6
    - ✓ Packets only fragmented at sending host. Will not be fragmented by any intermediate hosts.
    - ✓ Setting the IPV6_DONTFRAG socket option in the sending socket prevents the packet from being fragmented.

The Ping command is used to determine if a host is active. The Maximum Transmission Unit (MTU) is the largest size packet which can be sent on a network. Path MTU discovery is the process of dynamically determining the largest MTU value which can be used to send packets to a destination host without causing fragmentation of the packets.

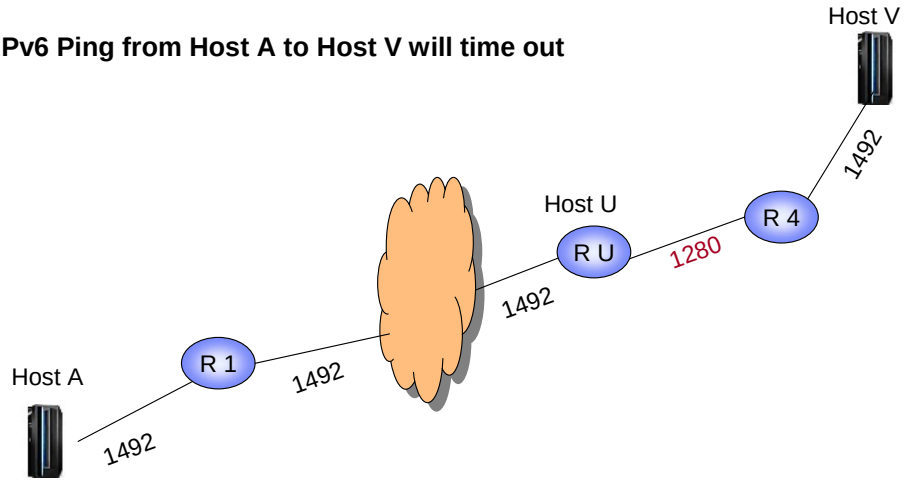
Problem statement - Need to determine network MTU values

- Detecting MTU problems in a network
  - Difficult to determine where MTU problem exists in a network
    - ✓ Fragmentation of packets hides problem
    - ✓ No z/OS command supports detecting where MTU problem exists in a network
  - Path MTU discovery helps avoid fragmentation
    - ✓ For IPv4, must be configured and is only triggered by TCP processing
    - ✓ Doesn't provide information about where problem exists in a network
- Determining current path MTU value
  - No command displays current path MTU value to a destination host

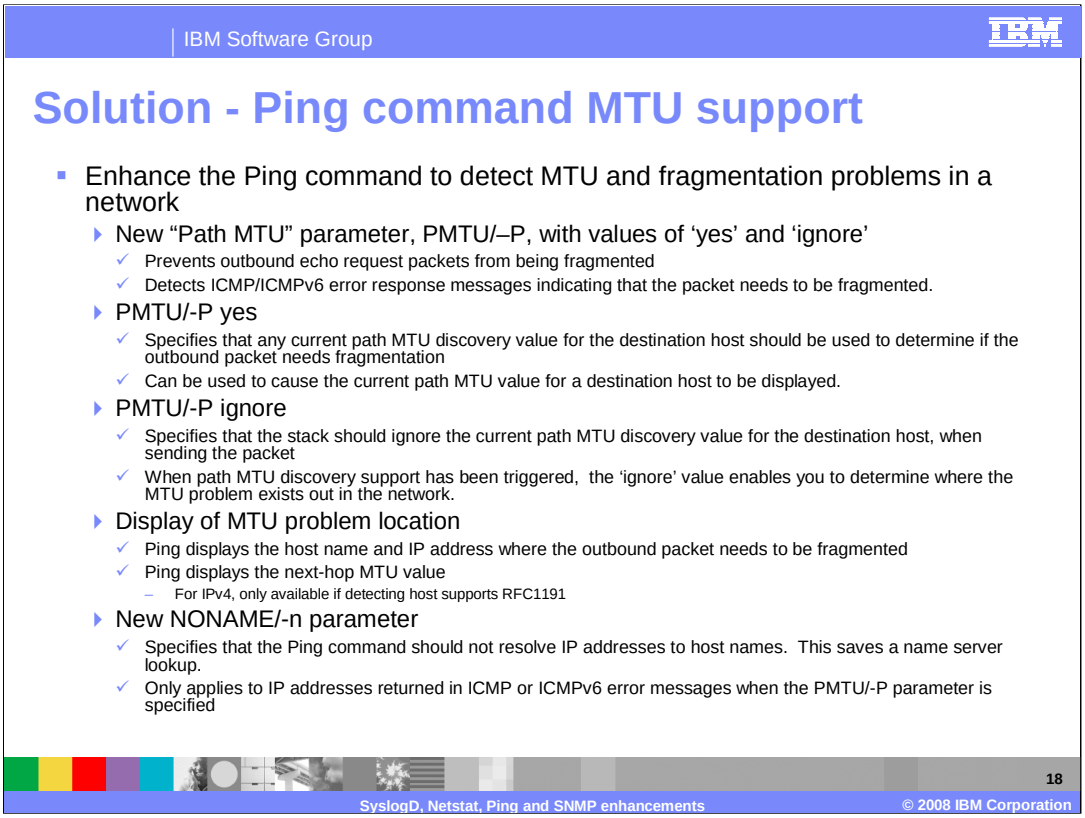
MTU problems can exist in large networks. The problem occurs where the MTU for a segment of the network is smaller then the network segments to which it is connected. Detecting MTU problems is difficult because, when the problem occurs, the IPv4 packets are normally fragmented. And z/OS CS currently does not provide any command to detect MTU problems. Path MTU discovery support can help avoid fragmentation by determining the smallest MTU value for the path to a destination host. But it does not provide information about where the problem is located in the network.

MTU problem in network

- **IPv4 Ping from Host A to Host V, with a length of 1400, will be fragmented by Host U**
- **IPv6 Ping from Host A to Host V will time out**



This slide shows an example of an MTU problem in a network. In this network, the segment from a router named Host U, to Router 4, has a smaller MTU than the other network segments. IPv4 Ping requests from Host A to Host V will be fragmented at Host U. IPv6 Ping requests from Host A to Host V will time out because intermediate IPv6 routers are not permitted to fragment packets. Only the sending TCP/IP stack is permitted to fragment outbound packets. The R1 and R4 figures represent other routers in the network.



A new method is provided in z/OS V1R9 Communications Server for determining where in the network an MTU problem exists, and for determining what the current path MTU value is to a destination host.

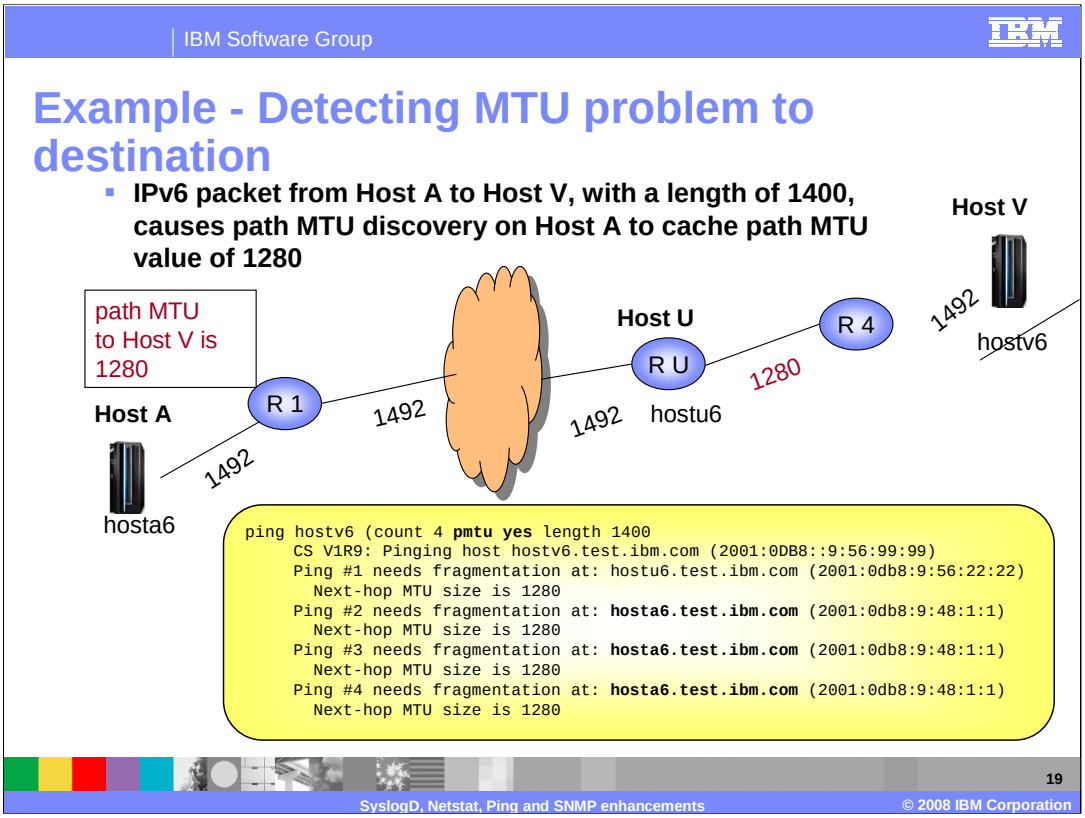
A new parameter, TSO PMTU or z/OS UNIX –P, has been added to the Ping command. Specifying this parameter prevents the outbound echo request packets from being fragmented. The values for this parameter are ‘yes’ and ‘ignore’.

A value of ‘yes’ specifies that the current path MTU value to the destination should be used to determine if the outbound packet needs to be fragmented. If path MTU discovery is not enabled, or has not been triggered, then the MTU associated with the outbound route is used to determine if the packet needs to be fragmented.

A value of ‘ignore’ specifies that any path MTU value to the destination should be ignored and, as long as the packet is not too large for the outbound interface, the packet should be sent out to the destination.

If the packet needs to be fragmented, either at the sending TCP/IP stack, or out in the network, the Ping command displays the host name and IP address of the host where fragmentation is needed. It also displays the next-hop MTU value returned by the host where fragmentation is needed. Hosts which support RFC1191 (Path MTU Discovery) should return the next-hop MTU value in the ICMP/ICMPv6 error message.

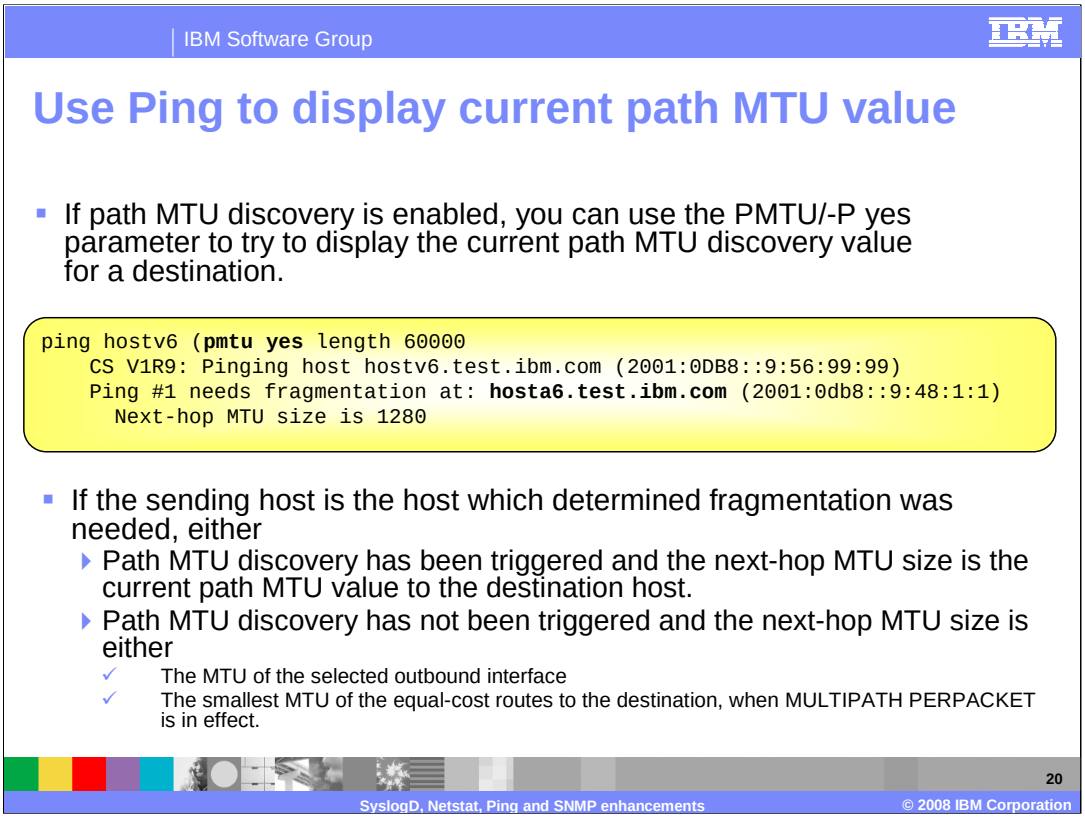
Another new parameter, TSO NONAME or z/OS UNIX –n, has been added to the Ping command. Specifying this parameter causes the Ping command to bypass the name server lookup of the host name, for the IP address of the host where fragmentation is needed. This Can be useful when the name server is slow or unresponsive or the environment does not support DNS reverse lookups (IP address to host name). This parameter is only in effect when the PMTU or –P parameter is specified.



This example shows the output from the Ping command which could have caused the path MTU value of 1280 to be set. Path MTU discovery is always enabled for IPV6 processing. So when Host U indicated that fragmentation was needed for packet #1, with a next-hop MTU size of 1280, this value was cached as the path MTU size to Host V.

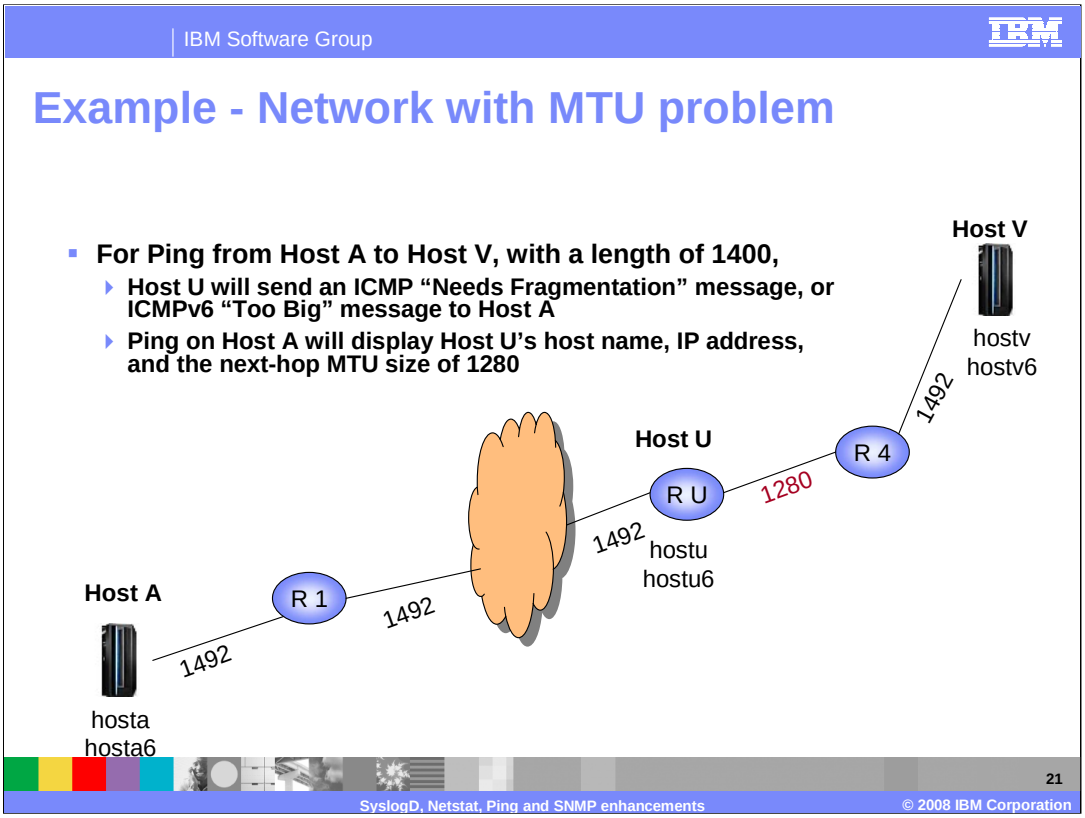
Since the **PMTU YES** parameter was specified on the Ping command, the outbound Ping packets will not be fragmented and the TCP/IP stack will use the path MTU value to compare against the packet length.

So, when the packet #2 is sent out, the TCP/IP stack on Host A compares the size of the packet to the cached path MTU value and fails the send request because the packet is too large to be sent without fragmenting it. This same error occurs for packets #3 and #4. The cached path MTU value is in effect for 20 minutes (assuming that, during that time, no other ICMPv6 error messages are received to decrease the path MTU size to Host V).



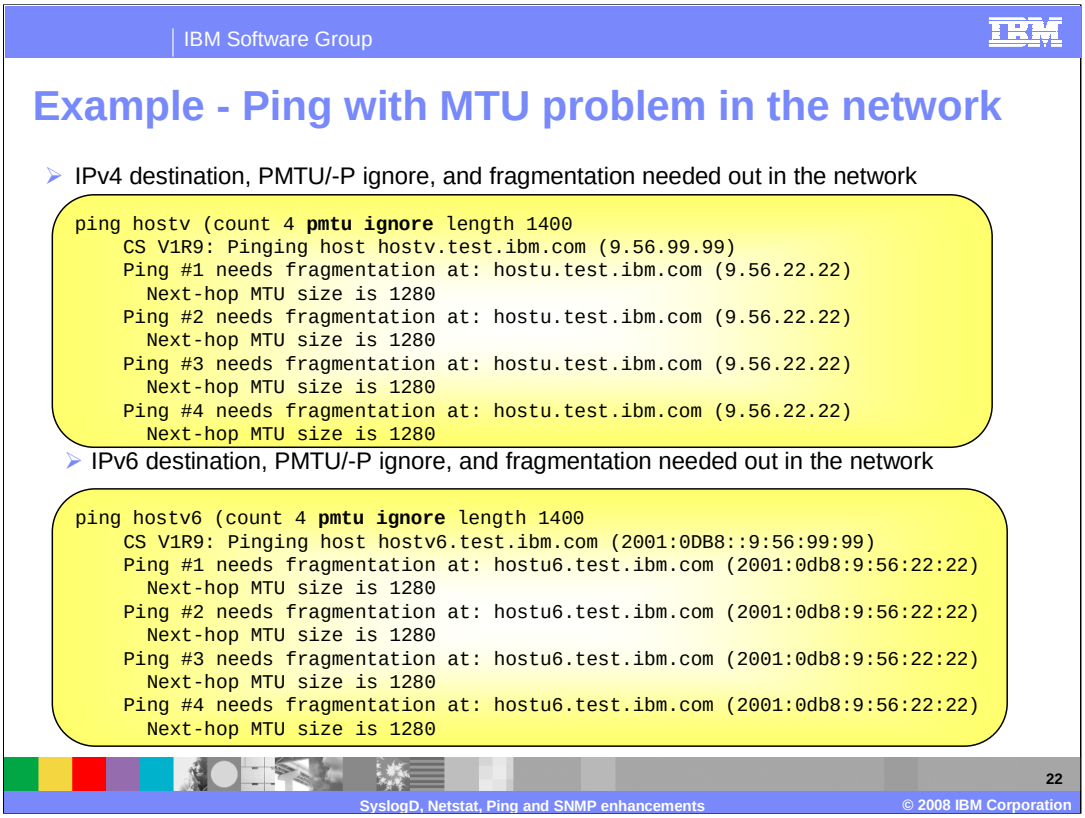
The Ping command with the **PMTU/-P YES** parameter, can enable you to display the cached path MTU discovery value for a destination host.

If you invoke the command, **ping hostv6 (pmtu yes length 60000)**, and the outbound packets need fragmentation at the sending TCP/IP stack, then the next-hop value displayed is either the path MTU value to the destination host, or the MTU of the outbound interface.



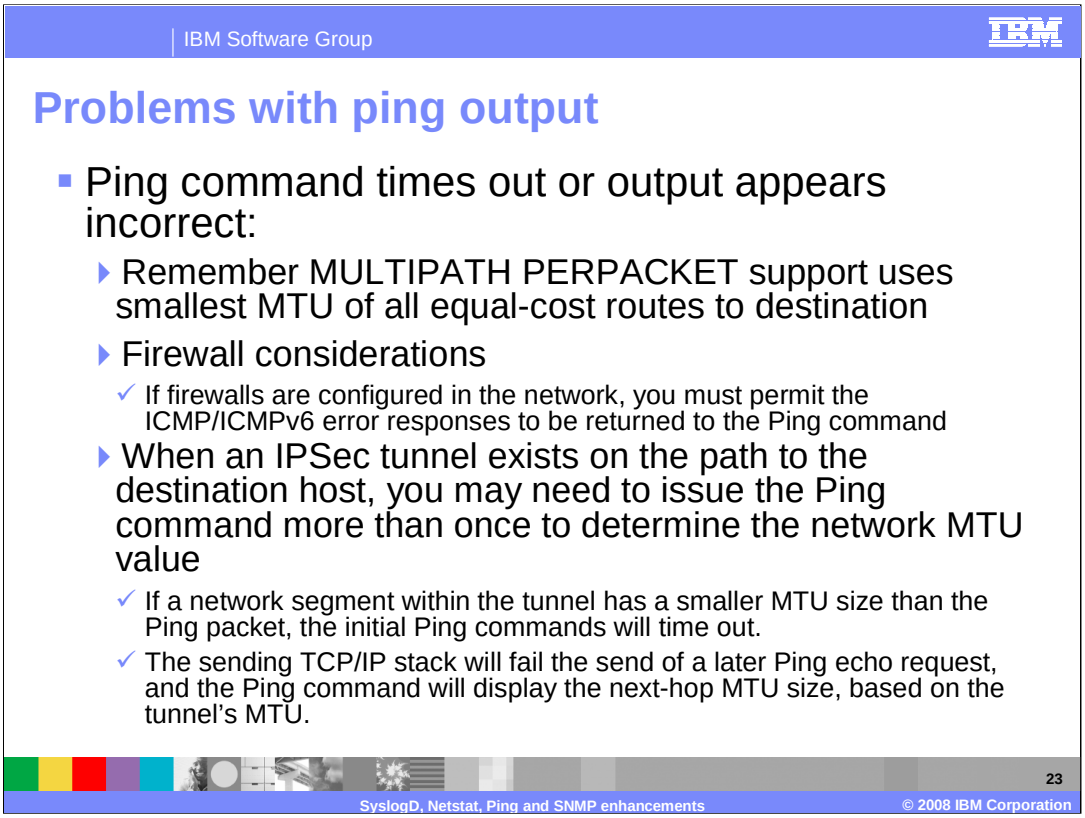
In this network, there is an MTU problem. To determine where the problem is, you can use the Ping command with the **PMTU/-P** parameter, so the outbound packets will not be fragmented. And since it is suspected that the problem is somewhere out in the network, **IGNORE** is specified for the PMTU/-P parameter so that any path MTU value cached at Host A for destination Host V will be ignored.

Since the packet data length is 1400, Host U could only forward the packet if it could fragment the packet. Instead Host U sends an ICMP or ICMPv6 error message back to Host A to indicate that the packet is too big and needs to be fragmented.



This example shows the output of the Ping command which was used to detect the network MTU problem in the network on the previous slide. The Ping command was invoked on Host A, with a destination host of Host V. Since it was suspected that the problem was out in the network somewhere, **PMTU IGNORE** was specified so that the packet would be sent out, even if path MTU discovery had determined that the path MTU size was smaller than the length of 1400.

The Ping responses show that the MTU problem is at Host U. And the next-hop MTU size from Host U to the network segment which leads to Host V is 1280.



This slide lists configuration options which can influence successful completion of the Ping command, and the Ping command output.

If firewalls are configured in the network, then you must permit ICMP/CMPv6 packets. The following ICMP/ICMPv6 error messages must be permitted to be returned to the Ping command:

ICMP

Type x'03' Destination unreachable

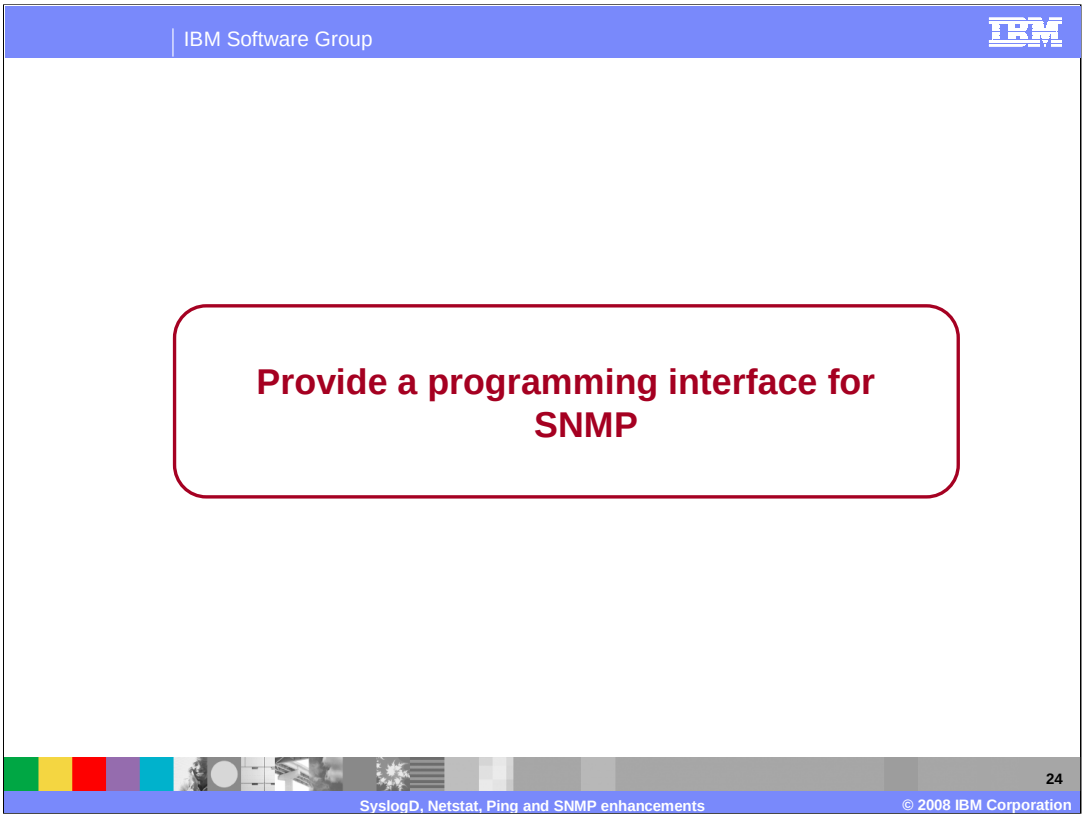
Code x'04' Fragmentation needed

ICMPv6

Type x'02' Packet too big

Code x'00'

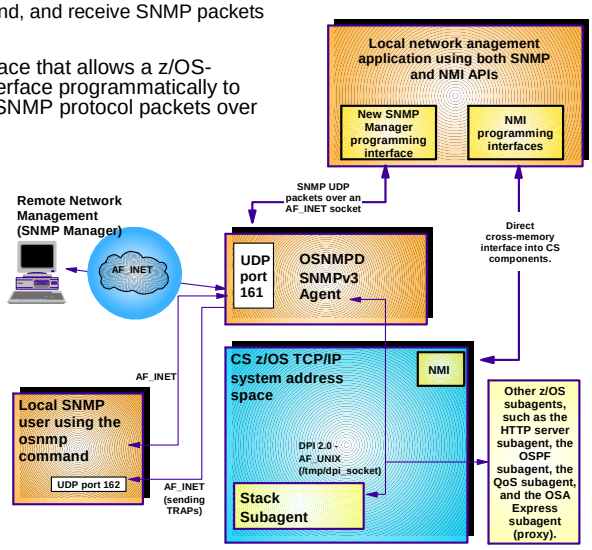
When an IPSec tunnel exists on the path to the destination host, the initial Ping commands may time out. For example, if a network segment in the tunnel has a smaller MTU than the Ping packet length, the first Ping echo request will time out because the ICMP/ICMPv6 error response will contain the encapsulated packet, not the original Ping echo request. This information cannot be correlated with the original echo request. On a subsequent Ping, the sending TCP/IP stack is aware of the path MTU information for the tunnel. So, subsequent Ping echo requests will fail, and the next-hop MTU size will then be displayed by the Ping command.



This section describes the SNMP Manager API and the SNMP Notification API, which can be used to build an SNMP Manager application.

Need a programming interface for the SNMP manager

- The osnmp command is a user interface to the SNMP protocol
  - Uses a built-in set of functions to build, send, and receive SNMP packets
- z/OS V1R9 provides a programming interface that allows a z/OS-resident SNMP manager application to interface programmatically to the SNMP agent (OSNMPD) exchanging SNMP protocol packets over a local TCP/IP socket
- Local management applications that need frequent access to high-volume management data should use the NMI interfaces:
  - NMI provides low-overhead access to high-volume management data
  - Typically addresses performance monitoring applications
- Local management applications that need access to SNMP management data that is not provided in NMI can use this new API:
  - Resource monitoring
  - Availability management
  - Operations (using SNMP SET)



SNMP is a standard's-based protocol for network management that is based upon the TCP/IP protocol (UDP). Part of the SNMP standards also includes a database structure specification for management objects called MIBs (Management Information Base). Use of SNMP is widespread, and work continues in the IETF mostly in the area of defining new MIBs. The SNMP protocol has been evolving for many years and has yielded several levels of the protocol, some of which were never adopted. Primarily, the supported protocol levels are: SNMPv1, SNMPv2c, and SNMPv3. SNMPv3 defines a user-based security model for SNMP, rather than the community-based model of SNMPv1 and SNMPv2c.

The key management entities of SNMP are:

•Agent - This entity implements the SNMP protocol stack and routes requests from managers to the appropriate subagents. Sometimes it is called the engine. It communicates with the subagents using the Distributed Protocol Interface (DPI) and with the Managers using the SNMP protocol. Subagents register their MIB objects with the Agent. For Comm Server, the agent is the osnmp daemon.

•Subagent - These entities are the providers of the MIB data. They communicate with the SNMP agents. In Comm Server, an example is the TCP/IP Subagent.

•Manager- These entities communicate using SNMP protocol requests with SNMP agents to retrieve management data. Comm Server only provides a command-line manager, the osnmp command. The manager function is typically part of management applications, such as Netview.

z/OS has not provided a formal, standards-based SNMP API for customer-driven manager applications. The osnmp command currently being shipped is, itself, a manager application, but customers have not been able to create their own application to manage SNMP, due to the absence of an external API.

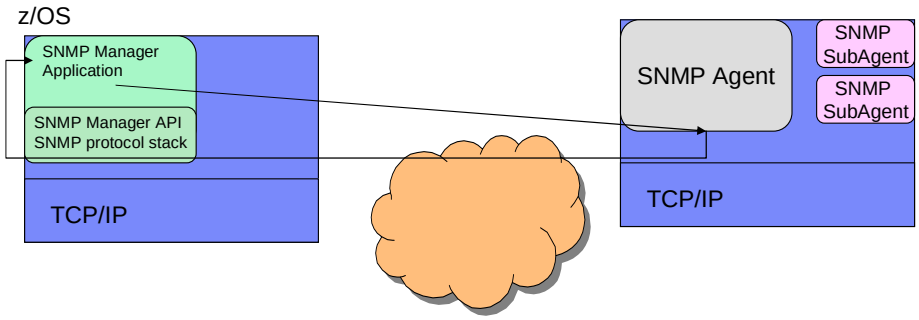
z/OS Communications Server V1R9 provides a new application programming interface (API), the SNMP Manager API, for writing SNMP managers. Management application developers can use this API to build SNMP management functions to retrieve SNMP management data. This API provides the following functions:

•The ability to build and send SNMP messages for SNMPv1, SNMPv2, and SNMPv3 and receive responses

•The ability to decode the SNMP messages and retrieve the SNMP data

z/OS Communications Server also provides an extension of the SNMP Manager API, the SNMP Notification API, which uses the functionality of the SNMP Manager API to send notifications to SNMP agents or SNMP Notification Receivers. Available notifications include Informs and both Version 1 and Version 2 Traps.

Solution - Provide a programming interface



- Provides a well-defined efficient API to build an SNMP Manager on z/OS
  - Provides the ability to build and send SNMP messages for SNMPv1, SNMPv2, and SNMPv3 and receive responses
  - Provides the ability to decode the SNMP messages and retrieve the SNMP data
  - Provides the ability to send notifications to SNMP agents or SNMP Notification Receivers
- Allows a z/OS management application to manage any other platform that supports SNMP

z/OS V1R9 Communications Server provides a new application programming interface (API), the SNMP Manager API, for writing SNMP managers. Management application developers can use this API to build SNMP management functions to retrieve SNMP management data.
z/OS V1R9 Communications Server is also providing an extension of the SNMP Manager API, the SNMP Notification API, which uses the functionality of the SNMP Manager API to send notifications to SNMP agents or SNMP Notification Receivers. Available notifications include Informs and both Version 1 and Version 2 Traps.
This API will allow z/OS management applications to manage any other platforms that support SNMP. This slide shows that an SNMP management application, running in z/OS, can manage any other platform that supports SNMP.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_SyslogD_Netstat_Ping_SNMP.ppt

This module is also available in PDF format at: ../SyslogD_Netstat_Ping_SNMP.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM z/OS

UNIX is a trademark of The Open Group in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.