HOT TOPICS

A z/OS Newsletter

In this issue...

your
designated server
drive the future

February 2004 - Issue 10

I can see clearly now: NetView for z/OS

BY LARRY GREEN AND KIM BAILEY

You're testing your new server code. The client is able to connect, but the server terminates with an error indicating that it received some incorrect data. You need a TCP/IP packet or data trace to see what the client is actually sending. It's been a while since you've had to collect and format a trace. So you'll need to find your notes on that. Let's see… Hmm… You have to enter some console commands to activate the SYSTCPDA trace component, and start the packet trace. After the trace is collected to a data set or from a dump of the TCP/IP dataspace, the traces must be formatted under IPCS. It's not so difficult, but with your schedule, you really don't have time for it all.

Wouldn't it be nice to have a tool that didn't require these tedious manual steps? Now you do. Using new function in NetView® for z/OS, you can quickly and easily capture and view formatted TCP/IP packet trace data in real time. When you need it. On demand.

NetView does the work for you!

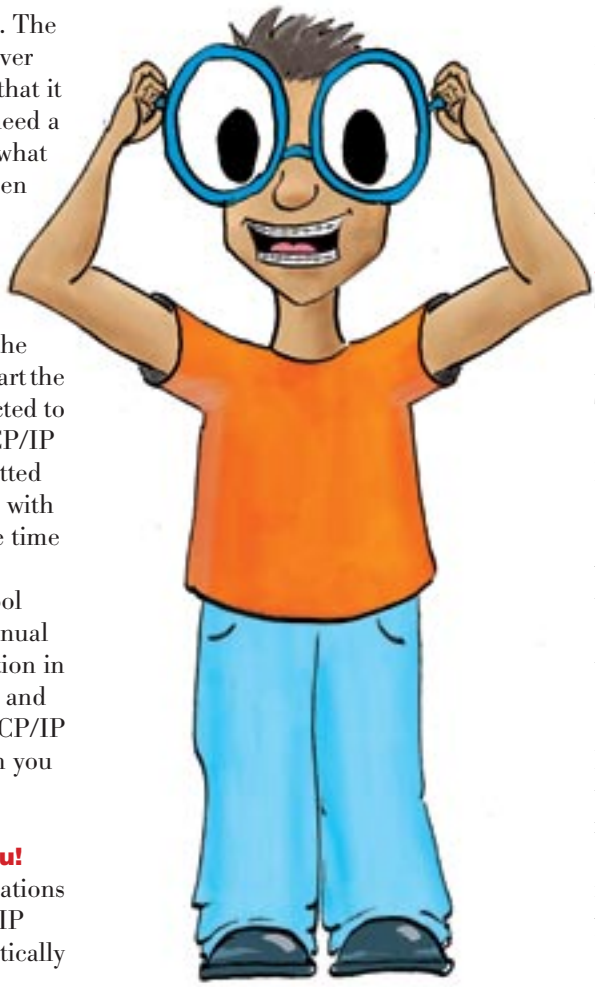
NetView uses a new z/OS Communications Server interface that provides TCP/IP packet or data trace data, programmatically and in real time, to management applications that provide packet trace services, such as formatting. New NetView trace functions allow you to specify the TCP/IP packets that you want to trace. Then, Netview collects the traces from the TCP/IP stack and provides the formatted data right to your screen.

How?

It's easy. NetView provides two new commands, **PKTS** and **FMTPACKT**, to capture and format the packet trace data collected by the TCP/IP stack. To get started, define an autotask to capture the data: `PKTS DEFINE OPID=`*`autotaskname`*

Then, start capturing data:
`PKTS START TCPNAME=`*`stackname`*

If you need to collect trace data on an ongoing basis, you can have these two steps done automatically whenever NetView starts up.



To do this, place the following statement in the NetView style sheet, CNMSTYLE:
`FUNCTION.AUTOTASK.PKTS.`*`stackname`* `=`
`autotaskname`

This statement associates an autotask with a stack. If you're interested in more than one stack, simply add another `FUNCTION` statement for it. For example:
`FUNCTION.AUTOTASK.PKTS.`*`stackname2`* `=`
`autotaskname2`

We expect that stacks will usually be specified in CNMSTYLE. However, if you need to change definitions on the fly without having to recycle NetView, the **PKTS DEFINE** command lets you do that, as shown earlier. The `INIT.PKTS=YES` statement starts NetView's data capture function.

Although NetView is doing all the work to make packet trace gathering and viewing easier, it's still the TCP/IP stack that collects the actual data. From the stack's perspective, collecting trace data has not changed.

Let's turn on tracing for the stack with the `TRACE CT,ON,COMP=SYSTCPDA, SUB=`*`jobname`* command, and tell the stack that you want packet trace data with the `VARY TCPIP,,PKTTRACE` command.

What do you get for your efforts?

The **PKTS QUERY** command lets you display the data you've captured. Only the interface names (LOOPBACK, TCPIPLINK) are readable. Most of the data is binary. It's not terribly useful to a human, but it is suitable for use by REXX execs to analyze and respond to problem conditions as part of your automation—which we expect to be the primary use for the **PKTS QUERY** command.

Using the default parameter values, the **PKTS QUERY** command reports on all TCP/IP stacks defined, and all addresses, ports, and interfaces. This might be more information than you need, so the **PKTS** command offers considerable flexibility in the area of filtering. For example, if you only want information related to a particular local address, you can limit the query output using a command like this: `WINDOW PKTS QUERY LADDR=12.34.56.789`. You can also filter on any combination of remote address, local or remote port, interface name, and time range.

For something more useful to human users, let's turn to the **FMTPACKT** command. Users familiar with Communications Server's packet trace facilities will recognize many of the parameters available and find that the trace formatting is the same as that under IPCS. (You don't have to learn to interpret a different trace format!) Using all the default parameter values, when you enter the **WINDOW FMTPACKT** command, you will see results like those shown on page 29.

The results show information about all defined stacks, all addresses and ports, TCP, UDP, and so on—everything that's been reported. (The command allowed the lines of this display to wrap so that all the information would be visible for our screen capture. For greater readability, we could have avoided wrapping by using the

LINESIZE keyword to set a longer line length.)

Like **PKTS**, the **FMTPACKT** command is very flexible. If you don't need to display everything, but only want, for example, summary statistical information for a particular local address and local port combination, use a command like this:

```
FMTPACKT QUERY LADDR=12.34.56.789
  LPORT=80 STATS=SUMMARY
```

This command generates statistical reports showing byte and packet counts for each protocol, listing the number of records, the first and last record numbers, and the first and last record times for the specified address and port.

If you need more detailed statistics, `STATS=DETAIL` provides the same information plus the number of records selected by record type, device type, jobname, linkname, and protocol number.

If you need detailed information for TCP and UDP sessions, use **FMTPACKT SESSION=DETAIL**. This provides link speed, data segment statistics, window statistics, information on data quantity and throughput, details on packets, including duplicate ACKs, out of order segments, and fragments, and much more. You can also use **FMTPACKT** as a NetView PIPE stage in a REXX exec to pass formatted packet trace data to automation.

Some TCP/IP configuration is required

In order for applications such as NetView to use the new packet trace interface, the z/OS Communications Server TCP/IP stack must be enabled by specifying the new **NETMONITOR PKTTRCSERVICE** configuration statement in the TCP/IP profile.

This ensures that network administrators allow the TCP/IP stack to provide this information to interfacing management applications. (Remember, tracing can affect system performance and the trace data may contain sensitive information.)

Additionally, optional security controls using RACF, or an equivalent security product, are provided to allow network administrators to restrict access to specific management applications.

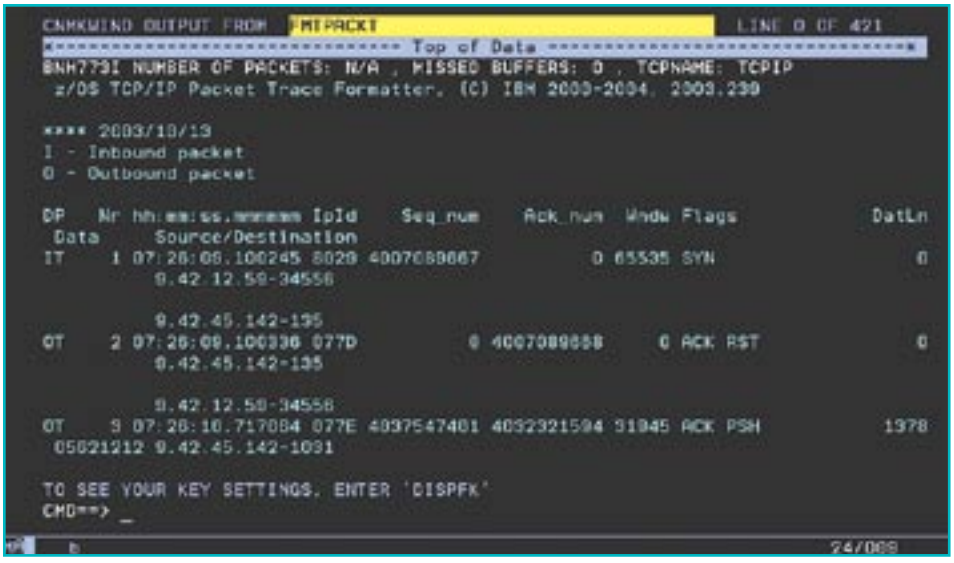
Multiple management applications can use the new TCP/IP packet trace interface simultaneously. When you do this, the currently active TCP/IP packet or data trace options affect all applications collecting this data. In addition, any **VARY TCPIP,,PKTTRACE** or **DATTRACE** console commands that modify the current trace options will affect the trace data that is being collected by management applications that use the new TCP/IP interface.

Try it!

Tivoli® NetView for z/OS and z/OS Communications Server work together to provide functions that make your job easier, helping you solve problems quickly, and improving your productivity. If you need formatted real-time packet traces, and you're tired of the multiple steps needed to obtain and view the data (that is, when you can find your notes reminding you what they are), why not try the new NetView packet trace function? We think you'll find it a valuable tool that's quick and easy to use.

You'll need:

- NetView for z/OS V5R1 with APAR OA04304
- Communications Server for z/OS V1R5, or z/OS V1R4 with APARs PQ77244, PQ77837 and PQ79566.



Sample output from the FMTPACKT command.