The Making of a Professional cTrace Packet Analyzer

Including a detailed look at the benefits of cTrace analysis and how AES CleverView for cTrace Analysis can accelerate and simplify TCP/IP network problem solving

Prepared by:
Brian J. Clausen (brianc@aesclever.com)

Published and Distributed by:
AES, 149 Commonwealth Drive, Menlo Park, CA 94025
(650) 617-2400
www.aesclever.com

The Making of a Professional cTrace Packet Analyzer

The Need for a Component Trace Packet Analysis Tool

Many large IT organizations managing SNA-TCP/IP transitions are finding that their ability to adequately support TCP/IP is challenging, since most of their technical expertise is still SNA-centric. As TCP/IP networks become increasingly complex Component Trace Analysis is vital, but is often difficult to handle in-house. Some companies no longer have the expertise needed to undertake component trace analysis. Others may choose to outsource component trace analysis - a costly and time-consuming option. This paper will first look at the history of cTrace Analysis and the urgent need for an analysis tool. It will then discuss the functionality needed in a professional Component Trace Analysis Tool.

Tracing Roots

VTAM/GTF trace readers were essential back in the late 70's to mid 90's. Numerous IT shops had embraced SNA connectivity and VTAM-based applications (fig. 1) allowing for (arguably) the first true, multi-user, remote access to host-based applications through a unified network of communications hardware and software. This, in itself, presented various connectivity and handshaking dilemmas. The applications themselves had to work in conjunction with SNA principles and VTAM definitions in order to provide proper screen presentation based on the 3270-terminal type, send and receive the data correctly, and so on, while still giving satisfactory performance.

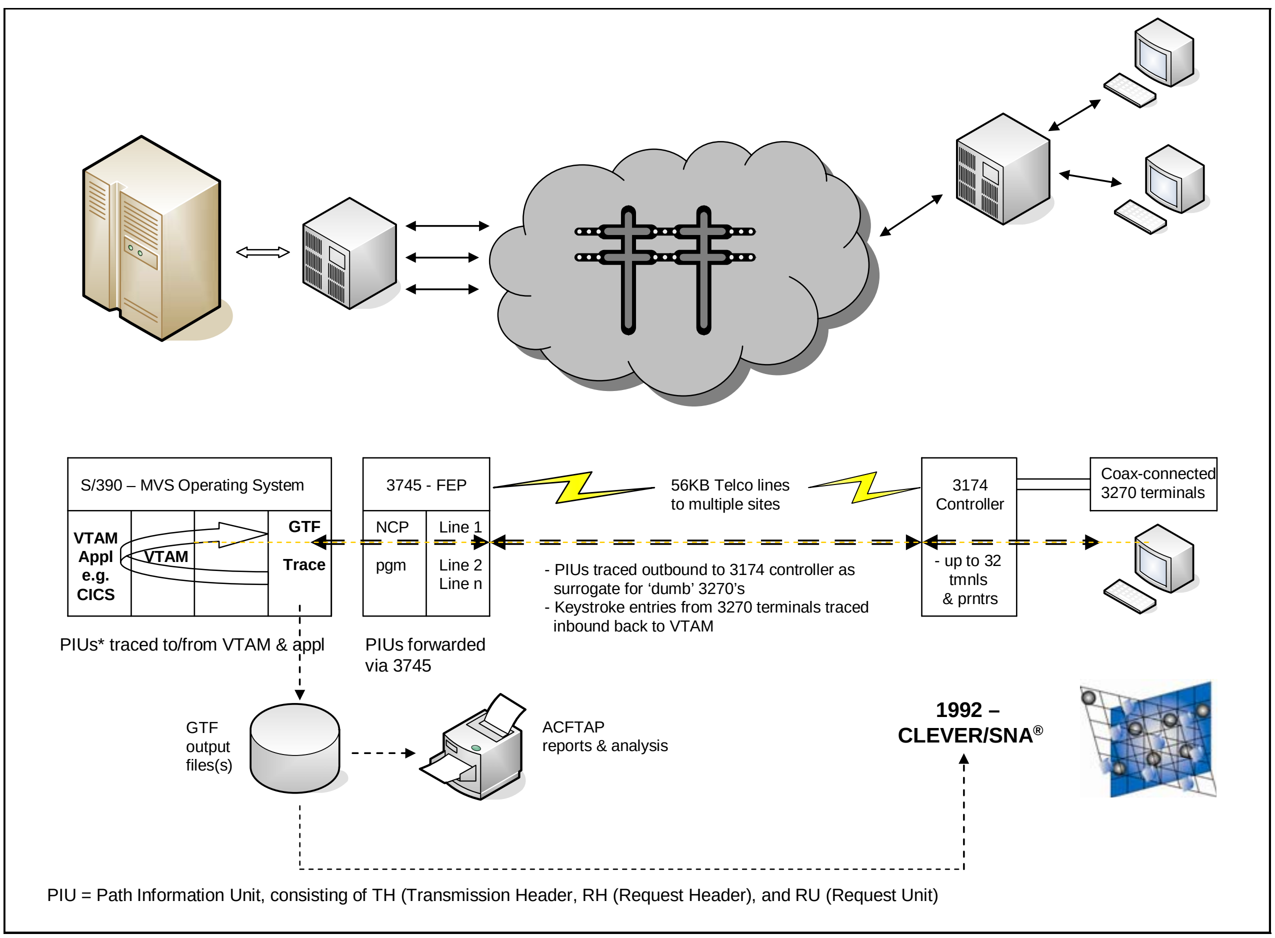


Figure 1. ***SNA-based topology and VTAM-based packet tracing and analysis choices***

The second generation of SNA, in the form of APPC and LU 6.2 sessions, allowed for more complex multi-peer conversations between an application and a single SNA connection or vice-versa. The rollout was even more complex. It was necessary to read subsequent traces to get the initial connectivity to work, and have a guarantee that it would continue to work when anything in the application or on the remote side was changed.

The Need for Speed

In the 80's the advent of T/R and Ethernet networks that operated ten to fifty times faster than their predecessors unveiled new protocols, applications, and subsequent troubleshooting of connectivity and performance. The early '90s saw the introduction of new, intelligent, PC-based tools that could automate trace capture and viewing. The most recognizable of these packet-capturing products was Sniffer™, which decoded TCP/IP and other LAN-based protocols, many of which have since been abandoned.

The emergence of TCP/IP as a bona fide standard occurred not just on LANs, but universally. This was true even of IBM's melded Communications Server, where VTAM has been repackaged as an end-point destination for blended network conversations. Reminders of VTAM's staying power include TN3270 sessions which are now pipelined directly via TCP/IP to the z/OS host. Alternatively, other VTAM-infused sessions are conducted via fourth-generation SNA links which in theory are a melding of APPN, HPR and UDP, but market-billed as Enterprise Extender. Depending on an application's stability, or its migration to TCP/IP, VTAM trace reading is still highly beneficial, but is not done as often. SNA Path Information Units (PIUs) are now embedded within the outer TCP/IP layers, making it more complex but no less pertinent to reveal hidden connectivity issues or the root cause for network latency.

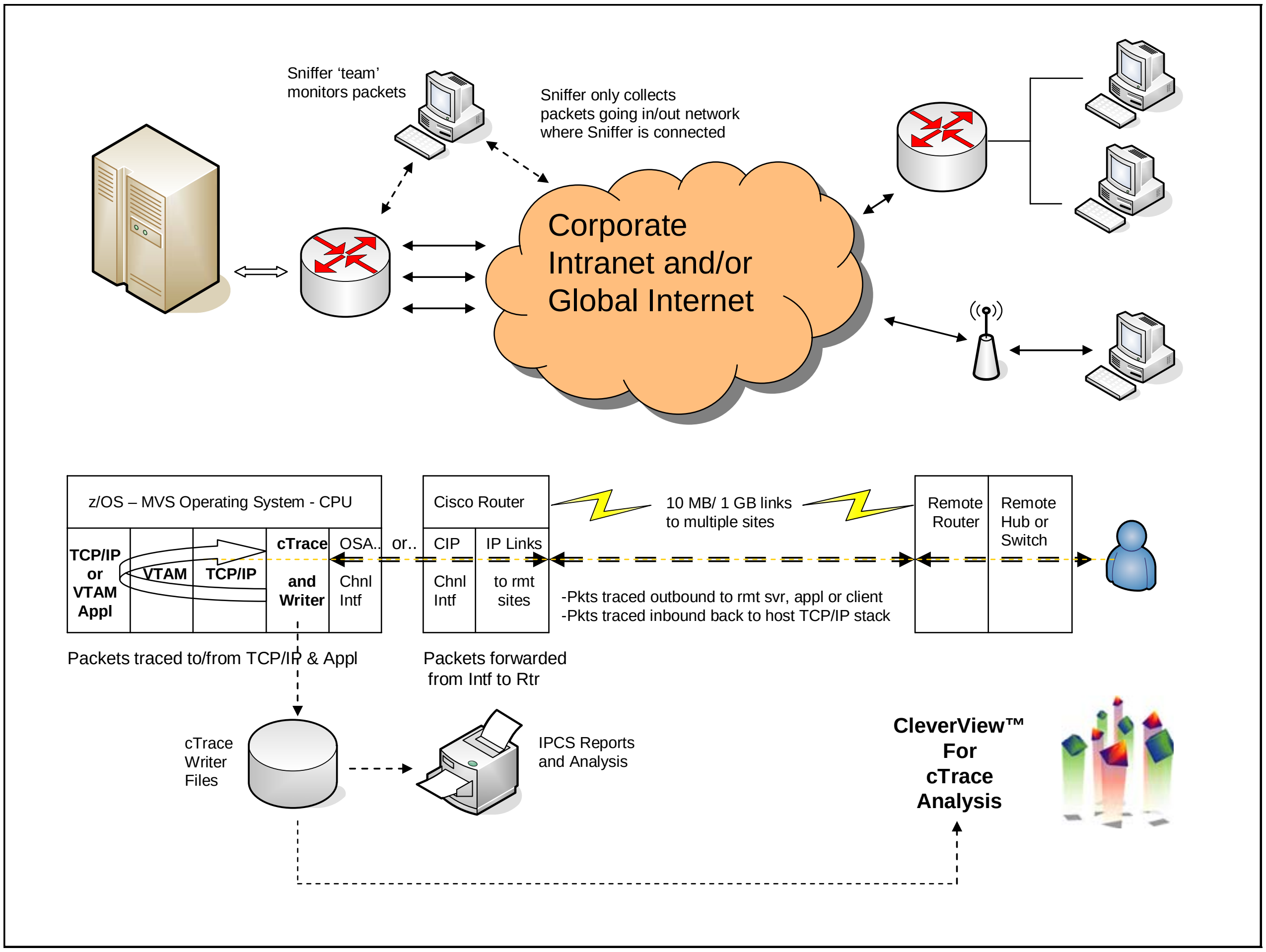


Figure 2. ***TCP/IP-based topology and TCP/IP packet tracing and analysis choices***

On the other hand, mission-critical legacy data bases are fronted by re-designed applications utilizing socket programming. These applications now rely fully on z/OS-based TCP/IP networks rather than the earlier SNA topology (fig. 2). Users are demanding an ever-increasing number of Service Oriented Applications (SOAs) that can enable instant asynchronous access to key data within multiple repositories, viewable via a browser-based GUI. Minimal downtime and 24x7 availability is mandated, requiring the ability to instantly diagnose ever more complex network problems while avoiding network outages.

The reality is that TCP/IP network problems may still arise, having a severe impact on Web, LAN-centric, and legacy mainframe applications. As complex issues with z/OS-based TCP/IP networks arise, technicians will have occasion to run component traces in order to isolate and resolve them. A network-centric diagnostic team could use information obtained from a Sniffer to determine their own world-view of traffic, but the information is often incomplete, lacking both sufficient evidence and the ability to see the z/OS TCP/IP stack side of traffic flows. Technicians need the ability to capture, decode, and further analyze TCP/IP packets coming into and out of the z/OS-based stack, applications, and/or IP channel interfaces.

IBM has provided a TCP/IP tracing mechanism called the Component Trace (cTrace), similar to VTAM/GTF traces, that can coexist with network-based Sniffers. The cTrace has many host-based data collection purposes, but it's most notable function is the collection of IP packets for subsequent analysis.

The Next Generation

The Component Trace is designed to capture diagnostic events and data for various components of the z/OS system. These various components are typically labeled with the prefix SYSxxxxx. To enable the collection of events the user needs to turn on a writer to record the events in a data file for subsequent analysis (fig. 3). To record events for one of several TCP/IP stack components, for example, the user would need to activate tracing by using the correct SYSTCPxx label (specifically, the SYSTCPDA component).

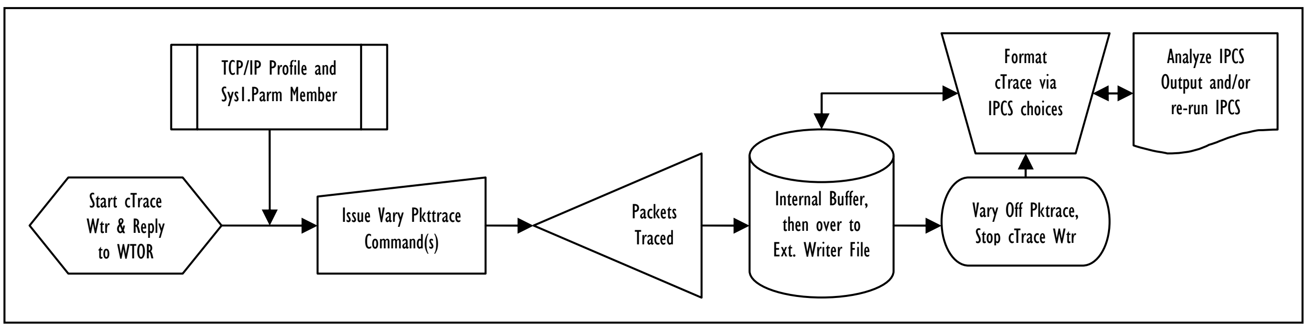


Figure 3. Standard cTrace Collection Process

TCP/IP packets are the first place to look for any issues related to connectivity, conversation errors, and/or performance issues (packet trace collection on the TCP/IP stack itself will also need to be activated in order to record these events). Nevertheless, analyses of these packets are the second line of defense in the isolation and/or elimination of the possibility of host-encountered TCP/IP issues; the first being the deployment of a good TCP/IP monitoring product. The third line of defense, usually the last resort, is the collection of additional stack component tracing in conjunction with third party (usually IBM) expert analysis of any particularly vexing stack or socket behavior problem.

Once the network technician enables the collection of specific packets via the TCP/IP PKTTRACE command, packets are copied to a temporary buffer via the SYSTCPDA component. An external Writer is used to redirect packets over to a permanent trace file. Specific parameters provide granular collection and/or filtering of packets by protocol type(s), by IP address(es), or by source and destination socket port numbers. The technician can choose one of three data collection options: the full packets, a truncated portion of each packet, or the data going in/out of a specific IP Channel interface only (usually OSAs or Cisco CIPs). The technician may also choose to record every packet, finding some way to filter and format the packets within the permanent file once the writer has been terminated.

This next generation of TCP/IP-packet cTrace reading on IBM mainframes can be very challenging. Raw cTraces are seemingly impossible to read and the basic tool (an enhanced version of Interactive Problem Control System - IPCS) provided by IBM can be cumbersome, time consuming, and difficult to interpret, even to the experienced. Given time and manpower constraints, honing trace reading skills stretches beyond the means of most IT organizations.

Given these constraints, technicians have sought a convenient, user-friendly and fresh approach to trace analysis that would allow them to streamline this tedious analysis process (fig 4.) and dramatically accelerate TCP/IP problem solving. This more automated approach would allow them to retain control of trace analyses, resolving most complex problems in-house, while also matching mandated business service-level objectives.

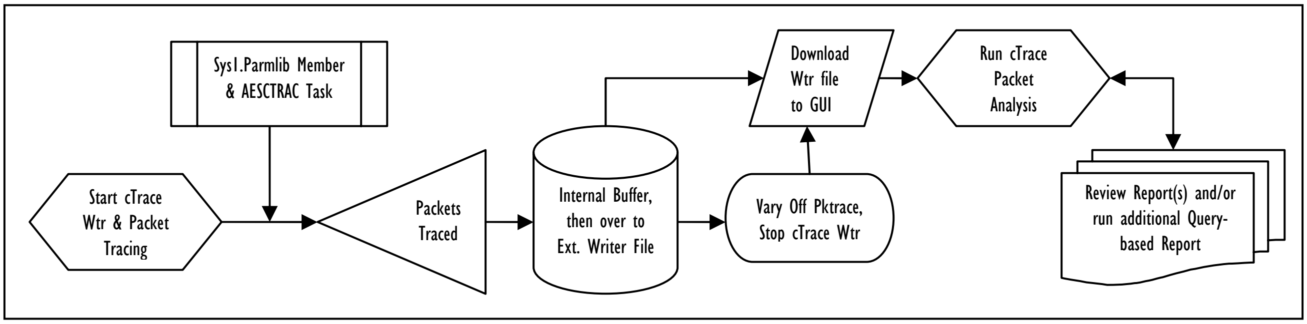


Figure 4. CleverView for cTrace Analysis Process

The Business Side

The objective of the IT organization is to keep all hardware, software, and interconnecting networks running for the rest of the corporation, and to provide the access required by their customers. For today's businesses to thrive, all aspects of TCP/IP functionality (including key TCP/IP services, applications, and routing) must run and communicate properly. Network systems professionals must identify and resolve TCP/IP outages or access issues as quickly as possible. TCP/IP wellness and performance under the z/OS-based umbrella is no exception.

In order to adequately maintain z/OS-based TCP/IP, technicians need a balance of three critical capabilities (fig 5.) available to them: monitoring, reporting, and diagnostics. It is this third criterion where cTrace analysis fits. Outages can neither be accurately predicted nor avoided, making the development of a smarter, quicker methodology for accurate analysis and exception identification essential.

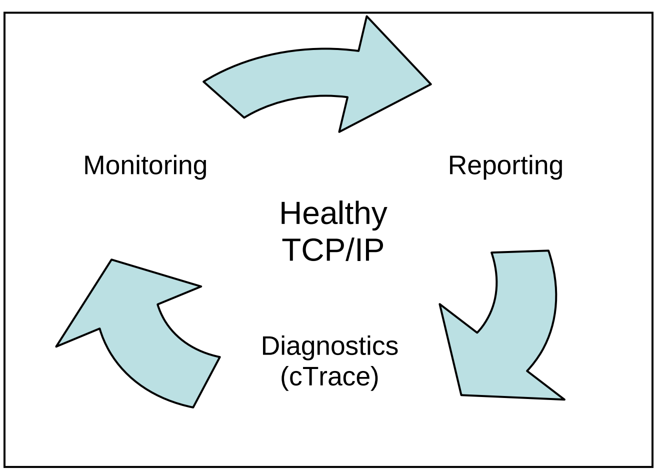


Figure 5. Three Essential Tools to Ensure TCP/IP Wellness

Many of the justifications for getting a TCP/IP monitoring and reporting product are equally applicable to a TCP/IP trace analysis and reporting tool, such as:

- Avoiding costs resulting from prolonged network or application outages
- Avoiding costs by identifying hidden issues that could lead to outages
- Saving both time and money by avoiding the need to outsource trace analysis
- Saving manpower by quickly ascertaining whether or not a problem is TCP/IP-related
- Tracking network service traffic and determining workload patterns for increased efficiency
- Correlating packet patterns/issues with network teams for departmental savings
- Assisting developers in profiling and tuning new applications for departmental savings
- Automating trace collection and analysis, saving both time and money
- Providing a knowledge base, user familiarity, and ease of training for departmental savings
- Eliminating excuses for not running the traces essential to root-cause discovery

The need for a GUI-based cTrace packet decoding and analysis tool is clear. The following pages provide detailed information about what goes into creating a Professional cTrace Packet Analyzer and explains the benefits of such a tool.

The Basic Requirements for a cTrace Packet Analyzer

A cTrace Packet Analyzer must be extremely efficient in its analyses, possess a dependable and thorough knowledge base, and must provide concise and accurate summary reports.

Core Elements

The cTrace Packet Analyzer must provide an easy, efficient, (and preferably automated) means (fig. 6) of starting the writer and issuing the needed commands. It should offer a venue for other potential user groups (such as Network Control Centers or Application Developers) to activate a cTrace, even though they may not be fully familiar with the nuances of TSO and ISPF-panel driven interfaces or possess a knowledge of parameters and their use. The interface should also have a simple method of saving trace parameters for future use, e.g. starting/stopping a trace, checking on the status of a trace, or transferring a trace.

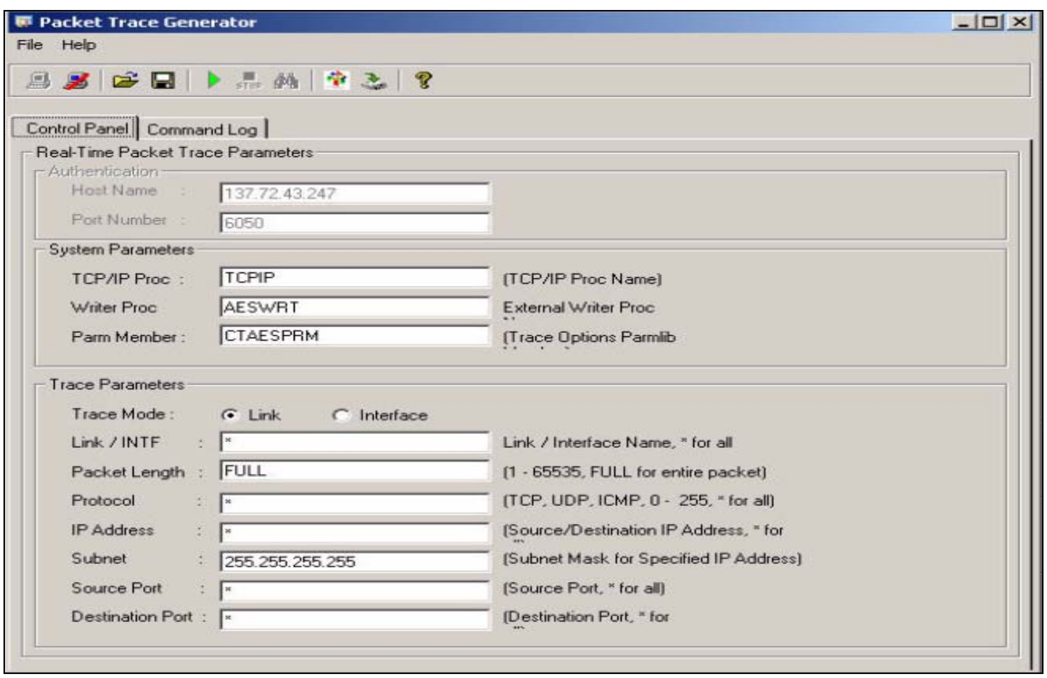


Figure 6. CleverView for cTrace Analysis Packet Trace Generator Panel

The cTrace Packet Analyzer should provide the flexibility to access this feature and perform the related functions from either the ISPF or GUI interface, without having to logon to TSO directly. It should allow the user to specify whether the trace is to be run for all packets or just for those packet characteristics and/or origination/destination(s) the user specifies. In addition, it should allow the user to initiate a cTrace on any LPAR on which the cTrace collector agent task is set up to run, not just on a single specific LPAR.

In order for cTrace analysis to really be useful in the long term, it is critically important to accumulate a knowledge base of network traffic flows and retain samples of specific scenarios and case experiences. It is therefore essential to have a product which provides the ability to create and maintain a cTrace database inventory. A ready-to-access cTrace inventory (fig. 7) will further accelerate the cTrace analysis process, by being able to identify, quantify, and manage the contents of these prior traces. It should also provide the ability to compare prior and current situations for, for example, time latency differences, specifically anticipated packet flows, or expected replies and responses.

Since the trace is run on the host and the intelligent analysis occurs on the workstation, the packet trace that must be transferred to the workstation via FTP or some other mechanism can be of considerable size. A cTrace Packet Analyzer with its own built-in FTP process, including a compression option, would address this challenge. Ideally, this file transfer and compression applet would also be available for other uses.

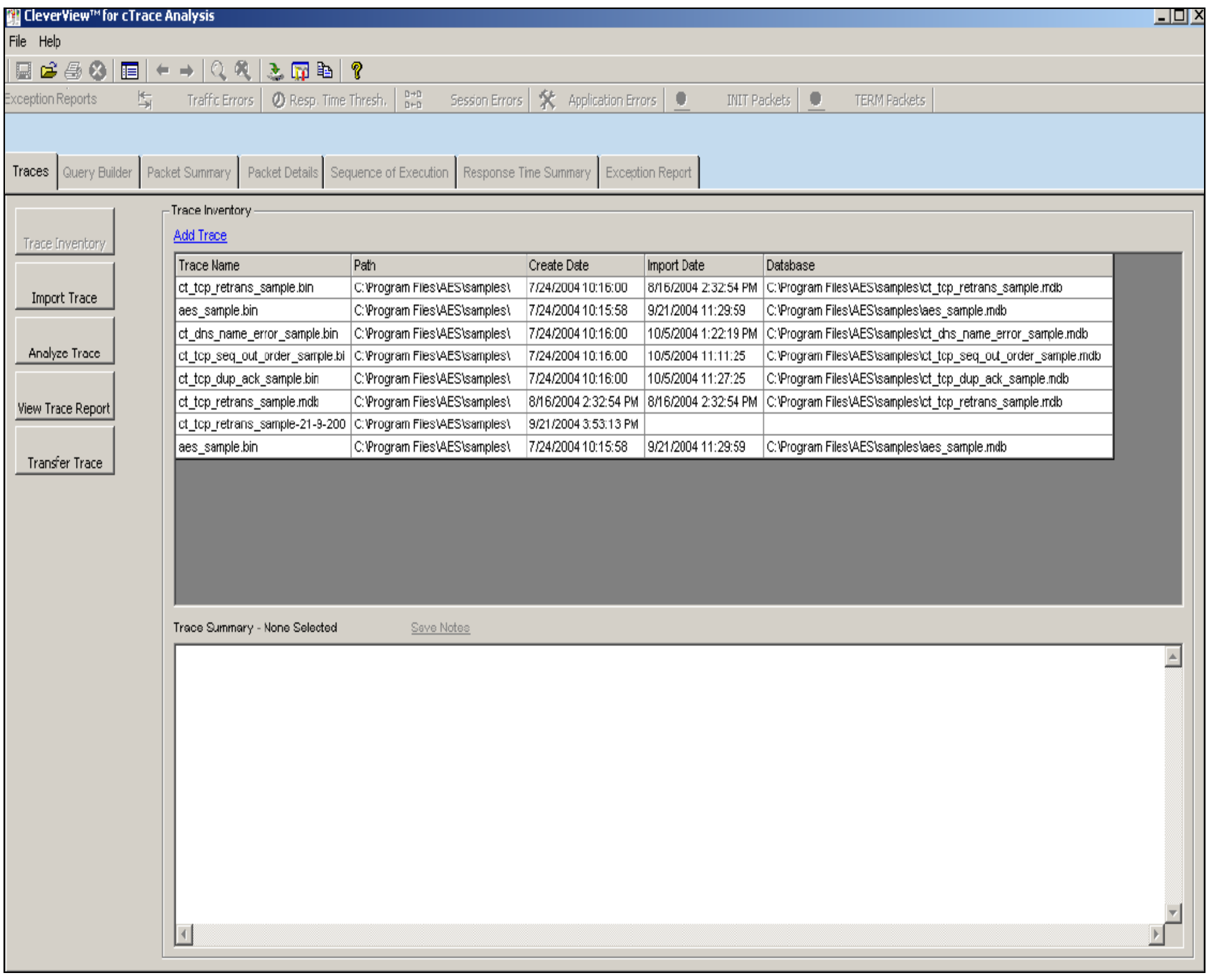


Figure 7. CleverView for cTrace Analysis Trace Inventory Table

This tool must have a dependable and thorough knowledge base. A cTrace Packet Analyzer should support and decode the entire array of possible TCP/IP-based protocols, including blended SNA-based traffic, such as for HPR/RTP conversations for Enterprise Extender links and, of course, TN3270 sessions. It should also support the latest advances of IP and ICMP V6.

The following protocols should be supported/decoded:

- EE/APPN Decoding: HPR/RTP, XID3s, FID5s, RHs, Control Vectors and GDS Variables
- Base Protocols: IP, TCP, UDP, IPv6
- Key Applications: Telnet, TN3270, TN3270E, FTP, LPR
- Routing: OSPF, RIPv1, RIPv2, EE/HPR
- Mail Protocols: SMTP, POP3
- Web Services: DNS, HTTP
- Management: SNMP, ICMP, ICMPv6
- Address Resolution: ARP, RARP, DHCP
- UNIX Remote Calls: RSH, REXEC, RLOGIN

It logically follows that the cTrace Packet Analyzer should include intelligent filters, enabling a user to create filtered query-based reports (fig. 8) for a specific subset of applications/protocols within selected record and date ranges, ports, IP addresses, and session criteria. This would streamline searches for specific timeframes, events, conversation socket(s), or protocol type(s), leading directly to the desired data for instant decoding and analysis.

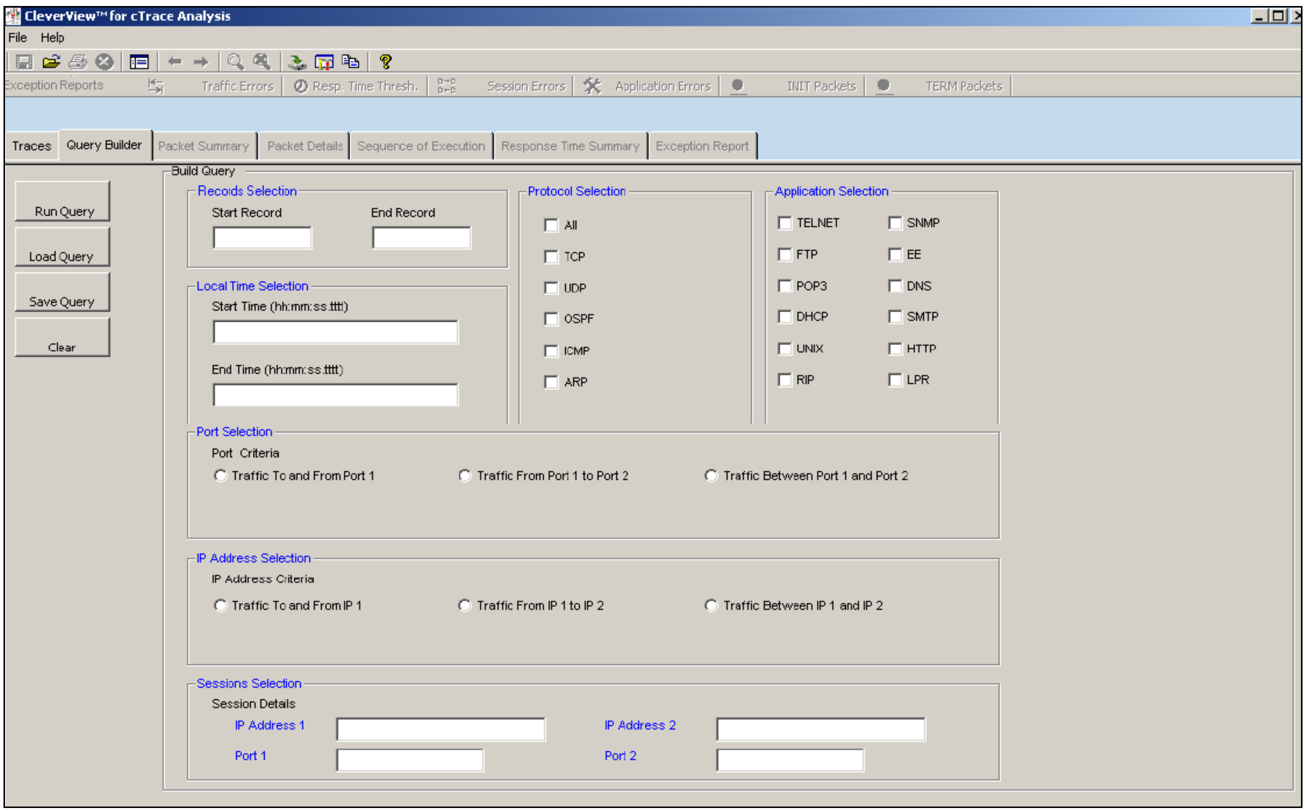
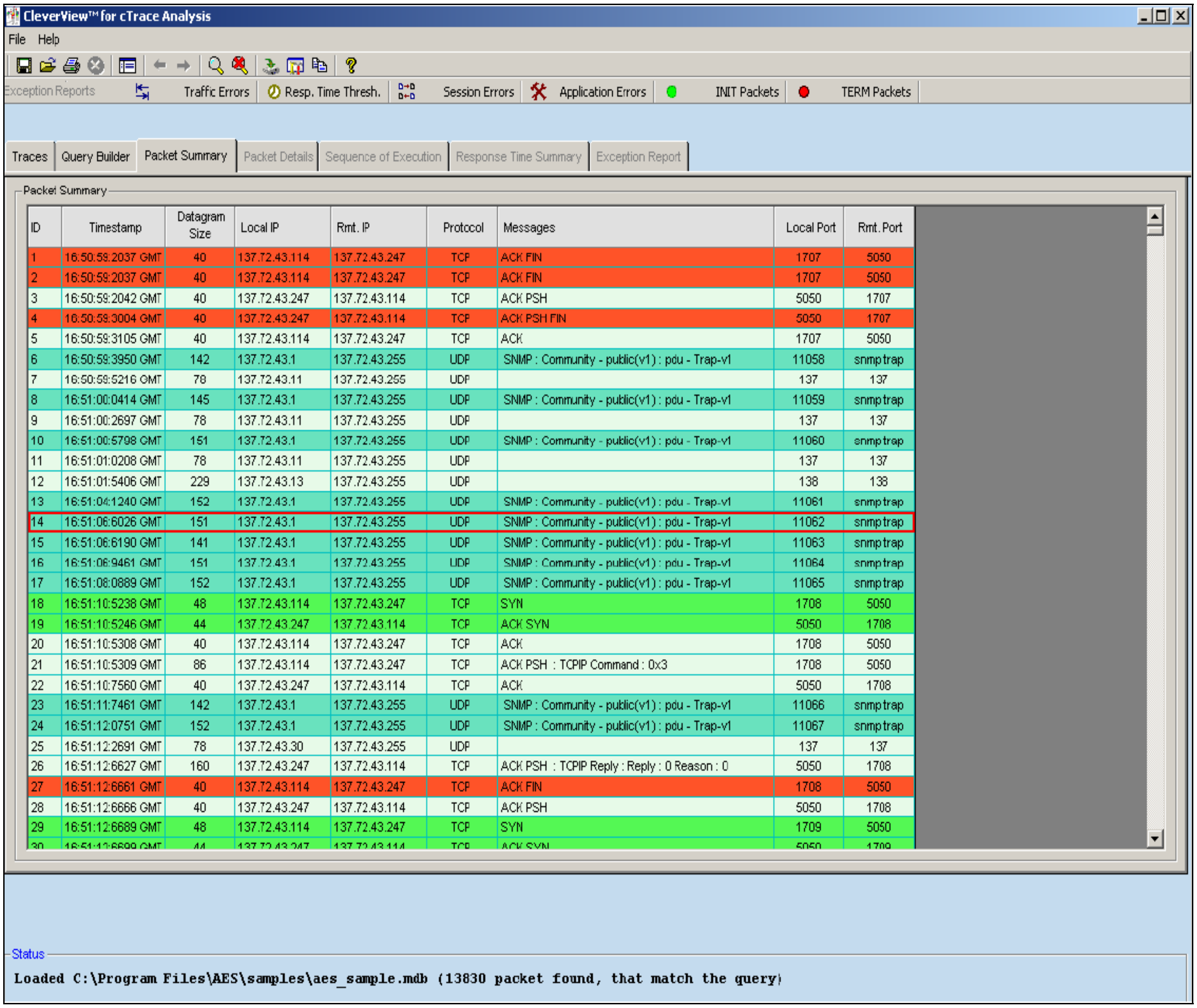


Figure 8. CleverView for cTrace Analysis Query Builder Choices

Packet Summary Reports

A packet summary report (*fig. 9*) is the best way to display the analysis of all packets in a trace file. If a filter is applied, only the packets that pass the filter should be displayed here. This is the stage at which diagnosis begins. Key fields should be identified in the summary list, including the packet number, timestamp, packet size, and local/remote (to/from) IP address and ports. It is also essential to identify the primary protocol of the packet and to summarize the packet's purpose based on its header information.



ID	Timestamp	Datagram Size	Local IP	Rmt. IP	Protocol	Messages	Local Port	Rmt. Port
1	16:50:59:2037 GMT	40	137.72.43.114	137.72.43.247	TCP	ACK FIN	1707	5050
2	16:50:59:2037 GMT	40	137.72.43.114	137.72.43.247	TCP	ACK FIN	1707	5050
3	16:50:59:2042 GMT	40	137.72.43.247	137.72.43.114	TCP	ACK PSH	5050	1707
4	16:50:59:3004 GMT	40	137.72.43.247	137.72.43.114	TCP	ACK PSH FIN	5050	1707
5	16:50:59:3105 GMT	40	137.72.43.114	137.72.43.247	TCP	ACK	1707	5050
6	16:50:59:3950 GMT	142	137.72.43.1	137.72.43.255	UDP	SNMP : Community - public(v1) : pdu - Trap-v1	11058	snmptrap
7	16:50:59:5216 GMT	78	137.72.43.11	137.72.43.255	UDP		137	137
8	16:51:00:0414 GMT	145	137.72.43.1	137.72.43.255	UDP	SNMP : Community - public(v1) : pdu - Trap-v1	11059	snmptrap
9	16:51:00:2697 GMT	78	137.72.43.11	137.72.43.255	UDP		137	137
10	16:51:00:5798 GMT	151	137.72.43.1	137.72.43.255	UDP	SNMP : Community - public(v1) : pdu - Trap-v1	11060	snmptrap
11	16:51:01:0208 GMT	78	137.72.43.11	137.72.43.255	UDP		137	137
12	16:51:01:5406 GMT	229	137.72.43.13	137.72.43.255	UDP		138	138
13	16:51:04:1240 GMT	152	137.72.43.1	137.72.43.255	UDP	SNMP : Community - public(v1) : pdu - Trap-v1	11061	snmptrap
14	16:51:06:6026 GMT	151	137.72.43.1	137.72.43.255	UDP	SNMP : Community - public(v1) : pdu - Trap-v1	11062	snmptrap
15	16:51:06:6190 GMT	141	137.72.43.1	137.72.43.255	UDP	SNMP : Community - public(v1) : pdu - Trap-v1	11063	snmptrap
16	16:51:06:9461 GMT	151	137.72.43.1	137.72.43.255	UDP	SNMP : Community - public(v1) : pdu - Trap-v1	11064	snmptrap
17	16:51:08:0889 GMT	152	137.72.43.1	137.72.43.255	UDP	SNMP : Community - public(v1) : pdu - Trap-v1	11065	snmptrap
18	16:51:10:5238 GMT	48	137.72.43.114	137.72.43.247	TCP	SYN	1708	5050
19	16:51:10:5246 GMT	44	137.72.43.247	137.72.43.114	TCP	ACK SYN	5050	1708
20	16:51:10:5308 GMT	40	137.72.43.114	137.72.43.247	TCP	ACK	1708	5050
21	16:51:10:5309 GMT	86	137.72.43.114	137.72.43.247	TCP	ACK PSH : TCPIP Command : 0x3	1708	5050
22	16:51:10:7560 GMT	40	137.72.43.247	137.72.43.114	TCP	ACK	5050	1708
23	16:51:11:7461 GMT	142	137.72.43.1	137.72.43.255	UDP	SNMP : Community - public(v1) : pdu - Trap-v1	11066	snmptrap
24	16:51:12:0751 GMT	152	137.72.43.1	137.72.43.255	UDP	SNMP : Community - public(v1) : pdu - Trap-v1	11067	snmptrap
25	16:51:12:2691 GMT	78	137.72.43.30	137.72.43.255	UDP		137	137
26	16:51:12:6627 GMT	160	137.72.43.247	137.72.43.114	TCP	ACK PSH : TCPIP Reply : Reply : 0 Reason : 0	5050	1708
27	16:51:12:6661 GMT	40	137.72.43.114	137.72.43.247	TCP	ACK FIN	1708	5050
28	16:51:12:6666 GMT	40	137.72.43.247	137.72.43.114	TCP	ACK PSH	5050	1708
29	16:51:12:6689 GMT	48	137.72.43.114	137.72.43.247	TCP	SYN	1709	5050
30	[illegible]	44	137.72.43.247	137.72.43.114	TCP	ACK SYN	5050	1709

Figure 9. CleverView for cTrace Analysis Packet Summary Report

Packet Detail Reports

A cTrace Packet Analyzer should provide details without requiring excessive time, undue effort, or added costs. It should be expected that for every summarized packet, the ability to zoom-in for a closer look at the entire recorded packet (*fig.10*) would also exist. Specifically, the packet break-out should be split into two parts: Packet Details and Hex Decode. Packet details should display the important fields from various headers of the packet and Request/Response Unit (RU) data, if any, should be captured. An option should also exist to display the RU data as either EBCDIC or ASCII. Hex Decode should display the contents of packets in hex format, broken down by header.

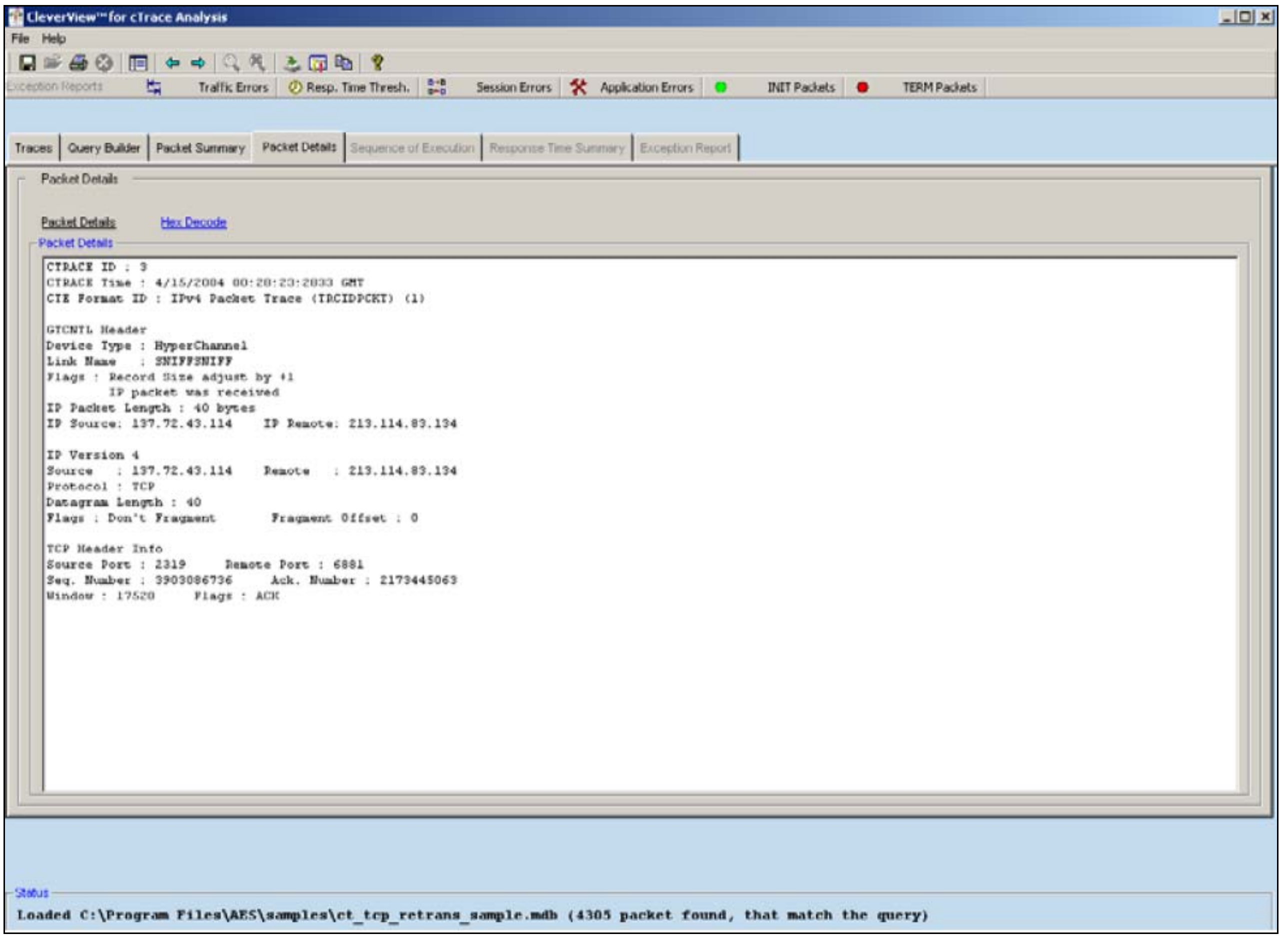


Figure 10. CleverView for cTrace Analysis Packet Details Report

Advanced Reporting

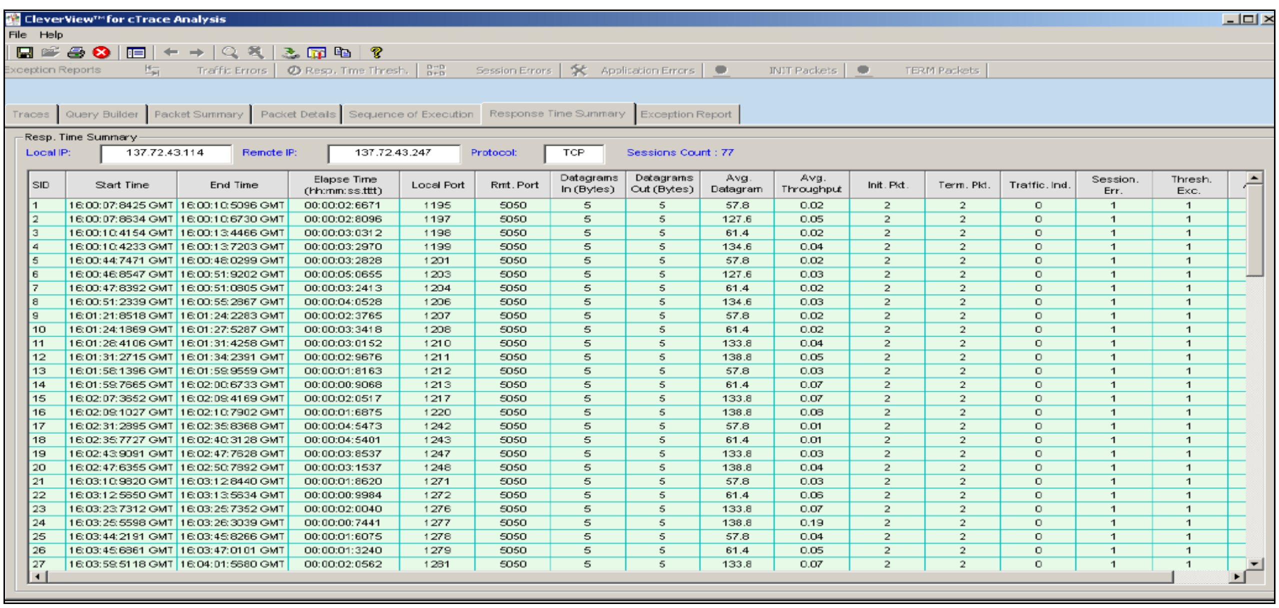
Basic queries, packet summaries, and detailed displays are a good starting point for any reliable cTrace Packet Analyzer, but to realize its fullest diagnostic potential, it must perform analysis equally well. The Analyzer should provide advanced reports that allow the user to quickly ascertain and address any performance latencies based on packet to/from directional flow and sequence analysis. It should offer the ability to view two cTrace packet reports side-by-side for comparative flow analysis. Lastly it should provide exception analysis, complimented by the ability to customize specific options in the reports, such as highlighting and threshold levels to flag errors.

To fully understand the advanced reporting requirement, a little background information may prove useful. In first-generation VTAM standards, a connection between a user's 3270 terminal and the host application was labeled a *session*. With APPC there could be multiple sessions between end-points, so IBM referred to each of these simultaneous sessions as *conversations* or *connections*. This latter name also applied to TCP/IP when its original specifications were created. Since there could be numerous simultaneous connections between two end-points using different TCP ports, they further delineated the term to refer to the combination of two IP Address partner pairs (the local and the remote, depending on your reference point), in combination with two allocated TCP port numbers (one local and one remote). This union is known as a *socket connection*.

This unique grouping of four numbers will usually be referred to as a *conversation* or a *session.* Note that all UDP traffic is described as *connectionless*, so although UDP traffic will be seen between two end-points, and perhaps consistently using the same two UDP ports on each end, it is still considered *conversation-less* because it is asynchronous in nature (i.e., any packet sent or received is independent of any other). In any case, whether dealing with TCP- or UDP-based packets and activity, the Analyzer should be able to quickly isolate and identify these communications between two IP addresses/port numbers using the same protocol. If the packet contains similar pairings of IP addresses/port numbers, they belong to the same socket connection, except as previously described for UDP packets. Without this foundation for connection awareness and packet trace filtration, then most advanced reporting capabilities described subsequently would be pointless.

Response Time Summary

Manually sorting and calculating response times from cTrace packets can be extremely tedious and highly error-prone. To accelerate these calculations, the Analyzer should summarize the response times (fig. 11) between a local IP interface and a remote IP target that are communicating via the same protocol. At a minimum, each session should be summarized in terms of how many packets were sent or received by the local IP address, the elapsed time of the entire session, the average throughput, and the average datagram size of each packet. Ideally, the report should include the number of INIT and TERM packets, Traffic Flow Indicator, the number of Session Errors, and whether any thresholds were exceeded.

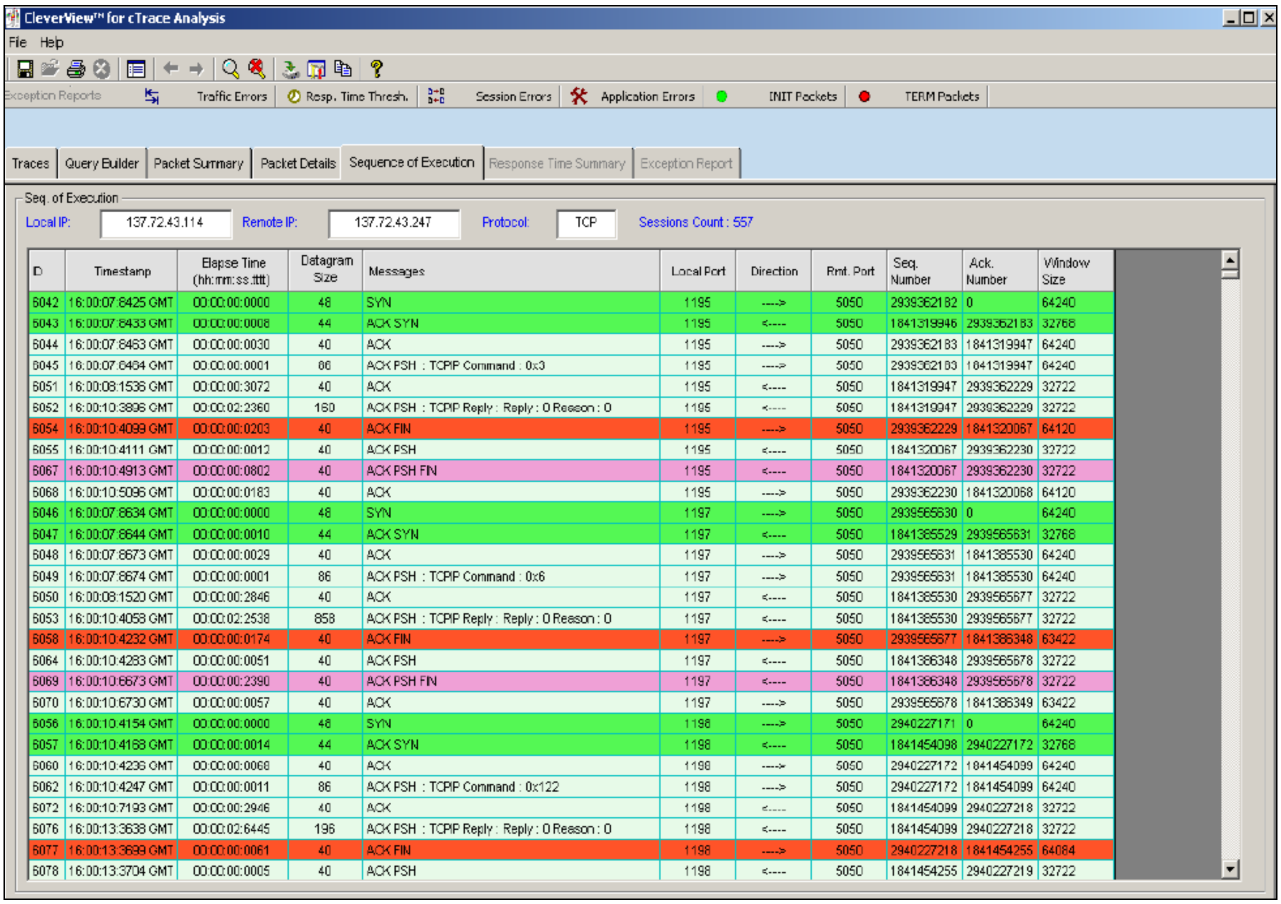


SID	Start Time	End Time	Elapse Time (hh:mm:ss.tttt)	Local Port	Rmt. Port	Datagrams In (Bytes)	Datagrams Out (Bytes)	Avg. Datagram	Avg. Throughput	Init. Pkt.	Term. Pkt.	Traffic. Ind.	Session. Err.	Thresh. Exc.
1	16:00:07:8425 GMT	16:00:10:5096 GMT	00:00:02:6671	1195	5050	5	5	57.8	0.02	2	2	0	1	1
2	16:00:07:8634 GMT	16:00:10:6730 GMT	00:00:02:8096	1197	5050	5	5	127.6	0.05	2	2	0	1	1
3	16:00:10:4154 GMT	16:00:13:4466 GMT	00:00:03:0312	1198	5050	5	5	61.4	0.02	2	2	0	1	1
4	16:00:10:4233 GMT	16:00:13:7203 GMT	00:00:03:2970	1199	5050	5	5	134.6	0.04	2	2	0	1	1
5	16:00:44:7471 GMT	16:00:48:0299 GMT	00:00:03:2828	1201	5050	5	5	57.8	0.02	2	2	0	1	1
6	16:00:46:8547 GMT	16:00:51:9202 GMT	00:00:05:0655	1203	5050	5	5	127.6	0.03	2	2	0	1	1
7	16:00:47:8392 GMT	16:00:51:0805 GMT	00:00:03:2413	1204	5050	5	5	61.4	0.02	2	2	0	1	1
8	16:00:51:2339 GMT	16:00:55:2867 GMT	00:00:04:0528	1206	5050	5	5	134.6	0.03	2	2	0	1	1
9	16:01:21:8518 GMT	16:01:24:2283 GMT	00:00:02:3765	1207	5050	5	5	57.8	0.02	2	2	0	1	1
10	16:01:24:1869 GMT	16:01:27:5287 GMT	00:00:03:3418	1208	5050	5	5	61.4	0.02	2	2	0	1	1
11	16:01:28:4106 GMT	16:01:31:4258 GMT	00:00:03:0152	1210	5050	5	5	133.8	0.04	2	2	0	1	1
12	16:01:31:2715 GMT	16:01:34:2391 GMT	00:00:02:9676	1211	5050	5	5	138.8	0.05	2	2	0	1	1
13	16:01:58:1396 GMT	16:01:59:9559 GMT	00:00:01:8163	1212	5050	5	5	57.8	0.03	2	2	0	1	1
14	16:01:59:7665 GMT	16:02:00:6733 GMT	00:00:00:9068	1213	5050	5	5	61.4	0.07	2	2	0	1	1
15	16:02:07:3652 GMT	16:02:09:4169 GMT	00:00:02:0517	1217	5050	5	5	133.8	0.07	2	2	0	1	1
16	16:02:09:1027 GMT	16:02:10:7902 GMT	00:00:01:6875	1220	5050	5	5	138.8	0.08	2	2	0	1	1
17	16:02:31:2895 GMT	16:02:35:8368 GMT	00:00:04:5473	1242	5050	5	5	57.8	0.01	2	2	0	1	1
18	16:02:35:7727 GMT	16:02:40:3128 GMT	00:00:04:5401	1243	5050	5	5	61.4	0.01	2	2	0	1	1
19	16:02:43:9091 GMT	16:02:47:7628 GMT	00:00:03:8537	1247	5050	5	5	133.8	0.03	2	2	0	1	1
20	16:02:47:6355 GMT	16:02:50:7892 GMT	00:00:03:1537	1248	5050	5	5	138.8	0.04	2	2	0	1	1
21	16:03:10:9820 GMT	16:03:12:8440 GMT	00:00:01:8620	1271	5050	5	5	57.8	0.03	2	2	0	1	1
22	16:03:12:5650 GMT	16:03:13:5634 GMT	00:00:00:9984	1272	5050	5	5	61.4	0.06	2	2	0	1	1
23	16:03:23:7312 GMT	16:03:25:7352 GMT	00:00:02:0040	1276	5050	5	5	133.8	0.07	2	2	0	1	1
24	16:03:25:5598 GMT	16:03:26:3039 GMT	00:00:00:7441	1277	5050	5	5	138.8	0.19	2	2	0	1	1
25	16:03:44:2191 GMT	16:03:45:8266 GMT	00:00:01:6075	1278	5050	5	5	57.8	0.04	2	2	0	1	1
26	16:03:45:6861 GMT	16:03:47:0101 GMT	00:00:01:3240	1279	5050	5	5	61.4	0.05	2	2	0	1	1
27	16:03:59:5118 GMT	16:04:01:5680 GMT	00:00:02:0562	1281	5050	5	5	133.8	0.07	2	2	0	1	1

Figure 11. CleverView for cTrace Analysis Response Time Summary Report

Sequence of Execution

A sequence of execution report (fig. 12) provides both a packet summary list and the packet details for all the packets exchanged between the source and destination host during any specific single (and perhaps multiple) subsequent socket connections within the cTrace collection of packets. This particular report should also highlight the initiation (shown in green) and termination (shown in red) for each of these sessions, as well as providing other specific event or threshold-defined highlighting. This makes it much easier to select specific elements from a mass of information.

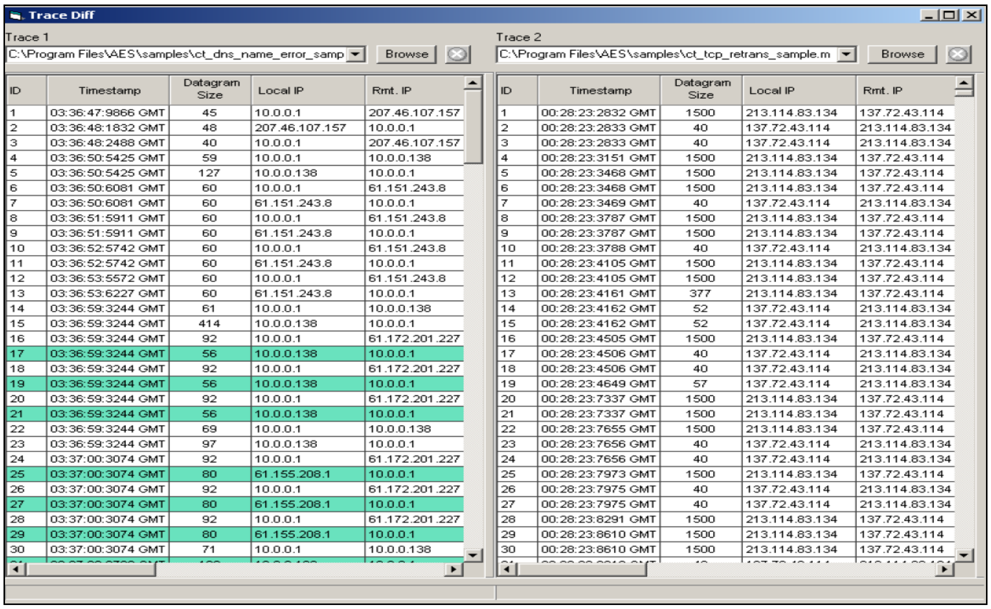


ID	Timestamp	Elapse Time (hh:mm:ss.tttt)	Datagram Size	Messages	Local Port	Direction	Rmt. Port	Seq. Number	Ack. Number	Window Size
6042	16:00:07.8425 GMT	00:00:00:0000	48	SYN	1195	---->	5050	2939362182	0	64240
6043	16:00:07.8433 GMT	00:00:00:0008	44	ACK SYN	1195	<----	5050	1841319946	2939362183	32768
6044	16:00:07.8463 GMT	00:00:00:0030	40	ACK	1195	---->	5050	2939362183	1841319947	64240
6045	16:00:07.8464 GMT	00:00:00:0001	86	ACK PSH : TCPIP Command : 0x3	1195	---->	5050	2939362183	1841319947	64240
6051	16:00:08.1536 GMT	00:00:00:3072	40	ACK	1195	<----	5050	1841319947	2939362229	32722
6052	16:00:10.3896 GMT	00:00:02:2360	160	ACK PSH : TCPIP Reply : Reply : 0 Reason : 0	1195	<----	5050	1841319947	2939362229	32722
6054	16:00:10.4099 GMT	00:00:00:0203	40	ACK FIN	1195	---->	5050	2939362229	1841320067	64120
6055	16:00:10.4111 GMT	00:00:00:0012	40	ACK PSH	1195	<----	5050	1841320067	2939362230	32722
6067	16:00:10.4913 GMT	00:00:00:0802	40	ACK PSH FIN	1195	<----	5050	1841320067	2939362230	32722
6068	16:00:10.5096 GMT	00:00:00:0183	40	ACK	1195	---->	5050	2939362230	1841320068	64120
6046	16:00:07.8634 GMT	00:00:00:0000	48	SYN	1197	---->	5050	2939565630	0	64240
6047	16:00:07.8644 GMT	00:00:00:0010	44	ACK SYN	1197	<----	5050	1841385529	2939565631	32768
6048	16:00:07.8673 GMT	00:00:00:0029	40	ACK	1197	---->	5050	2939565631	1841385530	64240
6049	16:00:07.8674 GMT	00:00:00:0001	86	ACK PSH : TCPIP Command : 0x6	1197	---->	5050	2939565631	1841385530	64240
6050	16:00:08.1520 GMT	00:00:00:2846	40	ACK	1197	<----	5050	1841385530	2939565677	32722
6053	16:00:10.4058 GMT	00:00:02:2538	858	ACK PSH : TCPIP Reply : Reply : 0 Reason : 0	1197	<----	5050	1841385530	2939565677	32722
6058	16:00:10.4232 GMT	00:00:00:0174	40	ACK FIN	1197	---->	5050	2939565677	1841386348	63422
6064	16:00:10.4283 GMT	00:00:00:0051	40	ACK PSH	1197	<----	5050	1841386348	2939565678	32722
6069	16:00:10.6673 GMT	00:00:00:2390	40	ACK PSH FIN	1197	<----	5050	1841386348	2939565678	32722
6070	16:00:10.6730 GMT	00:00:00:0057	40	ACK	1197	---->	5050	2939565678	1841386349	63422
6056	16:00:10.4154 GMT	00:00:00:0000	48	SYN	1198	---->	5050	2940227171	0	64240
6057	16:00:10.4168 GMT	00:00:00:0014	44	ACK SYN	1198	<----	5050	1841454098	2940227172	32768
6060	16:00:10.4236 GMT	00:00:00:0068	40	ACK	1198	---->	5050	2940227172	1841454099	64240
6062	16:00:10.4247 GMT	00:00:00:0011	86	ACK PSH : TCPIP Command : 0x122	1198	---->	5050	2940227172	1841454099	64240
6072	16:00:10.7193 GMT	00:00:00:2946	40	ACK	1198	<----	5050	1841454099	2940227218	32722
6076	16:00:13.3638 GMT	00:00:02:6445	196	ACK PSH : TCPIP Reply : Reply : 0 Reason : 0	1198	<----	5050	1841454099	2940227218	32722
6077	16:00:13.3699 GMT	00:00:00:0061	40	ACK FIN	1198	---->	5050	2940227218	1841454255	64084
6078	16:00:13.3704 GMT	00:00:00:0005	40	ACK PSH	1198	<----	5050	1841454255	2940227219	32722

Figure 12. CleverView for cTrace Analysis Sequence of Execution Report

Trace Comparison

Should a specific problem trace prove challenging to analyze further, it may be beneficial to compare that trace to a similar event trace where such a problem does not occur in order to examine the differences. With large traces, such comparisons can prove extremely difficult. The ability to compare two different trace analyses side by side (*fig. 13*) is an extremely powerful feature for a cTrace Packet Analyzer.



Trace Diff

Trace 1: C:\Program Files\AES\samples\ct_dns_name_error_samp

ID	Timestamp	Datagram Size	Local IP	Rmt. IP
1	03:36:47:9866 GMT	45	10.0.0.1	207.46.107.157
2	03:36:48:1832 GMT	48	207.46.107.157	10.0.0.1
3	03:36:48:2488 GMT	40	10.0.0.1	207.46.107.157
4	03:36:50:5425 GMT	59	10.0.0.1	10.0.0.138
5	03:36:50:5425 GMT	127	10.0.0.138	10.0.0.1
6	03:36:50:6081 GMT	60	10.0.0.1	61.151.243.8
7	03:36:50:6081 GMT	60	61.151.243.8	10.0.0.1
8	03:36:51:5911 GMT	60	10.0.0.1	61.151.243.8
9	03:36:51:5911 GMT	60	61.151.243.8	10.0.0.1
10	03:36:52:5742 GMT	60	10.0.0.1	61.151.243.8
11	03:36:52:5742 GMT	60	61.151.243.8	10.0.0.1
12	03:36:53:5572 GMT	60	10.0.0.1	61.151.243.8
13	03:36:53:6227 GMT	60	61.151.243.8	10.0.0.1
14	03:36:59:3244 GMT	61	10.0.0.1	10.0.0.138
15	03:36:59:3244 GMT	414	10.0.0.138	10.0.0.1
16	03:36:59:3244 GMT	92	10.0.0.1	61.172.201.227
17	03:36:59:3244 GMT	56	10.0.0.138	10.0.0.1
18	03:36:59:3244 GMT	92	10.0.0.1	61.172.201.227
19	03:36:59:3244 GMT	56	10.0.0.138	10.0.0.1
20	03:36:59:3244 GMT	92	10.0.0.1	61.172.201.227
21	03:36:59:3244 GMT	56	10.0.0.138	10.0.0.1
22	03:36:59:3244 GMT	69	10.0.0.1	10.0.0.138
23	03:36:59:3244 GMT	97	10.0.0.138	10.0.0.1
24	03:37:00:3074 GMT	92	10.0.0.1	61.172.201.227
25	03:37:00:3074 GMT	80	61.155.208.1	10.0.0.1
26	03:37:00:3074 GMT	92	10.0.0.1	61.172.201.227
27	03:37:00:3074 GMT	80	61.155.208.1	10.0.0.1
28	03:37:00:3074 GMT	92	10.0.0.1	61.172.201.227
29	03:37:00:3074 GMT	80	61.155.208.1	10.0.0.1
30	03:37:00:3074 GMT	71	10.0.0.1	10.0.0.138

Trace 2: C:\Program Files\AES\samples\ct_tcp_retrans_sample.m

ID	Timestamp	Datagram Size	Local IP	Rmt. IP
1	00:28:23:2832 GMT	1500	213.114.83.134	137.72.43.114
2	00:28:23:2833 GMT	40	137.72.43.114	213.114.83.134
3	00:28:23:2833 GMT	40	137.72.43.114	213.114.83.134
4	00:28:23:3151 GMT	1500	213.114.83.134	137.72.43.114
5	00:28:23:3468 GMT	1500	213.114.83.134	137.72.43.114
6	00:28:23:3468 GMT	1500	213.114.83.134	137.72.43.114
7	00:28:23:3469 GMT	40	137.72.43.114	213.114.83.134
8	00:28:23:3787 GMT	1500	213.114.83.134	137.72.43.114
9	00:28:23:3787 GMT	1500	213.114.83.134	137.72.43.114
10	00:28:23:3788 GMT	40	137.72.43.114	213.114.83.134
11	00:28:23:4105 GMT	1500	213.114.83.134	137.72.43.114
12	00:28:23:4105 GMT	1500	213.114.83.134	137.72.43.114
13	00:28:23:4161 GMT	377	213.114.83.134	137.72.43.114
14	00:28:23:4162 GMT	52	137.72.43.114	213.114.83.134
15	00:28:23:4162 GMT	52	137.72.43.114	213.114.83.134
16	00:28:23:4505 GMT	1500	213.114.83.134	137.72.43.114
17	00:28:23:4506 GMT	40	137.72.43.114	213.114.83.134
18	00:28:23:4506 GMT	40	137.72.43.114	213.114.83.134
19	00:28:23:4649 GMT	57	137.72.43.114	213.114.83.134
20	00:28:23:7337 GMT	1500	213.114.83.134	137.72.43.114
21	00:28:23:7337 GMT	1500	213.114.83.134	137.72.43.114
22	00:28:23:7655 GMT	1500	213.114.83.134	137.72.43.114
23	00:28:23:7656 GMT	40	137.72.43.114	213.114.83.134
24	00:28:23:7656 GMT	40	137.72.43.114	213.114.83.134
25	00:28:23:7973 GMT	1500	213.114.83.134	137.72.43.114
26	00:28:23:7975 GMT	40	137.72.43.114	213.114.83.134
27	00:28:23:7975 GMT	40	137.72.43.114	213.114.83.134
28	00:28:23:8291 GMT	1500	213.114.83.134	137.72.43.114
29	00:28:23:8610 GMT	1500	213.114.83.134	137.72.43.114
30	00:28:23:8610 GMT	1500	213.114.83.134	137.72.43.114

Figure 13. CleverView for cTrace Analysis TraceDiff Report

Exception Reporting

Ideally, a cTrace Packet Analyzer should automatically highlight common errors, and even some that are not so common. At the very least, the following errors should be pinpointed:

Traffic Errors — Packets that indicate traffic problems. Retransmission and congestion indicators would be included within this category.

Session Errors — Packets that indicate something is wrong with the current session. These are mainly transport layer errors.

Threshold Errors — Packets that exceed user-defined threshold settings within the option panel for the product.

Application Errors — Packets in which errors happen at the application level (fig. 14), such as FTP, DNS, DHCP, and SNMP.

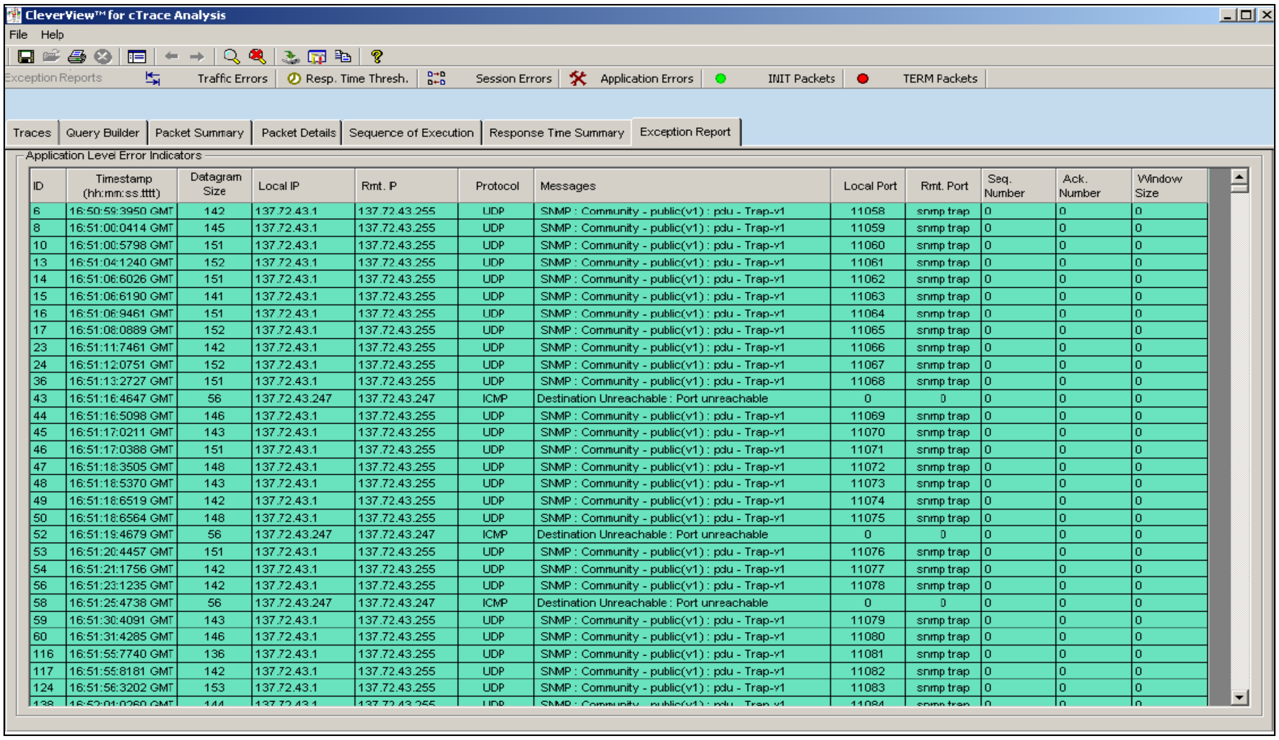


Figure 14. CleverView for cTrace Analysis Exception Report: Application Errors

Report Customization

In order to make the trace report exceptions and other pertinent data easier to read, different color highlighting can be used to differentiate packet events. This basic improvement can significantly accelerate and simplify cTrace analysis. For instance, the basic INIT (Initiation) and TERM (Termination) events can be highlighted in Green and Red as logical highlighting color defaults. These events are highlighted not just for TCP sessions, but for other conversation starts and ends, such as for EE link activations and terminations.

Another basic improvement with great benefit involves the cTrace data output format for timestamps. The default is GMT, requiring the user to make mental calculations depending where the packets were traced when they were reported and analyzed, further complicating the process. The Analyzer can handle this extra step instead by providing a customizable way to uniformly change the timestamps within each packet of a specific Analyzer report grouping. Ideally, a cTrace Packet Analyzer should also provide other customization options, as well as the capability to add more options in the future.

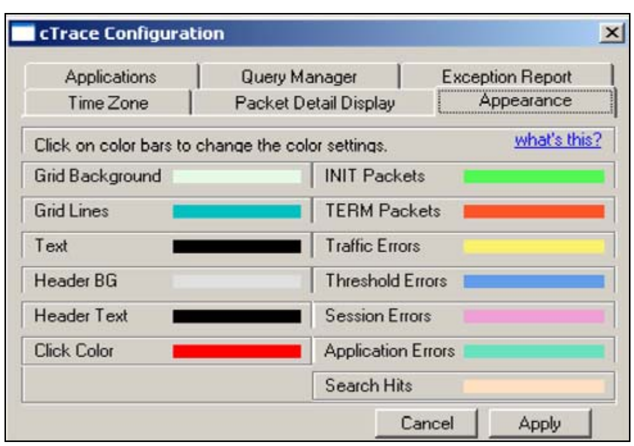


Figure 15. CleverView of cTrace Analysis Report Configuration Settings

Experience, Extras and Enhancements

An effective cTrace Packet Analyzer cannot remain static. Protocols for TCP/IP-based applications are constantly being enhanced. IBM is providing annually updated capabilities for the z/OS-based TCP/IP stack domain - capabilities that can even help to improve or add to the existing foundation of cTrace analysis and reporting. It is essential, that a professional cTrace Packet Analyzer tool will provide annual enhancements and updates, such as new reports or threshold highlighting. Ultimately, it is the combination of industry experience, extras, and enhancements which hold the key to making and maintaining a truly professional cTrace Packet Analyzer.

Real-Time Packet Tracing

IBM has included a new API within the z/OS V1R5 Communications Server that allows packet trace entries from the TCP/IP stack to be retrieved and viewed in real time, as they are being collected. To maximize the benefits available from this new functionality, a cTrace Analyzer should provide the capability to control all aspects of collecting, reviewing, and analyzing a component trace via an ISPF-based interface while logged onto a TSO session.

There could be a myriad of packets traveling between conversation pairs on TCP/IP at any one time. Attempting to digest all of these packets on a scrolling TSO screen would be a daunting task, if not totally overwhelming. Even so, there might very well be certain situations in which running a cTrace in real time would be advantageous. While a user needs to determine when and where this applies, it is recommended to run a real-time trace in conjunction with the filters offered by the Analyzer. The time and activity during these specific real-time collections must be controlled as best as possible, since there can be inordinate amounts of packet trace data collected in real time. In principle, the data space where packets are retained should be defined on the host by the user, so the Analyzer should use a wrap-around setup (fig. 16). This would mean that the quantity and quality of the real-time packet collection captured would be largely dependent on its defined size, though interactive analysis of the packets currently stored could be done repeatedly.

```
********************************** Top of Data ***********************************
 z/OS TCP/IP Packet Trace Formatter, (C) IBM 2000-2003, 2003.349

OPTIONS((Both Bootp(67,68) Cleanup(500) DelayAck(200,200) Domain(53)
 Dump(0) Finger(79) Flags() Format(Detail) Ftp(20,21) Gain(125,250)
 Gopher(70) Limit(999999999) Local Ntp(123) Option Noreassembly Router(520)
 Rpc(111) Nosegment Smtp(25) Snmp(161,162) Speed(10,10) Telnet(23) Tftp(69)
 Time(37) Userexit() Www(80)
  ))

**** 2005/03/11
RcdNr Sysname  Mnemonic Entry Id   Time Stamp      Description
----- -------- -------- -------- --------------- ------------------------------

--------------------------------------------------------------------------------
    1 OS15     PACKET   00000004 10:30:13.953912 Packet Trace
 From Interface     : ETH1                Device: LCS Ethernet     Full=318
  Tod Clock         : 2005/03/11 10:30:13.953909
  Sequence #        : 0                   Flags: Pkt
 IpHeader: Version  : 4                   Header Length: 20
  Tos               : 00                  QOS: Routine Normal Service
  Packet Length     : 318                 ID Number: 99E7
  Fragment          :                     Offset: 0
  TTL               : 127                 Protocol: UDP            CheckSum: FBA6 FF
  Source            : 137.72.43.222
  Destination       : 239.255.255.250

 UDP
  Source Port       : 1900  ()            Destination Port:  1900  ()
  Datagram Length   : 298                 CheckSum: 028F FFFF

IP Header           : 20
000000 4500013E 99E70000 7F11FBA6 89482BDE  EFFFFFFA

Protocol Header     : 8
000000 076C076C 012A028F

Data                : 290    Data Length: 290

--------------------------------------------------------------------------------
```

Figure 16. Example of Real-Time Tracing Decodes

Security and cTraces

The ability to protect confidential packet data should be an inherent part of the cTrace Packet Analyzer, and based on the foundations of z/OS constructs. In other words, the packet contents would be just as secure as the assigned writer file restrictions and user ID access. Use of specific product functions, whether it's the GUI or an ISPF-based panel, should be based on explicit user authorities and strict product licensing. Since the packets would be viewable for particular content, as has always been the case with Sniffer or GTF traces, cTrace Packet Analyzer efforts should only be assigned to trusted individuals with appropriate clearance levels.

There are few better ways to improve business security than to have the ability to trace and look for specific TCP/IP access breaches or attempts to cause harm to systems or data. The weekly scheduling of coordinated collections and analyses of TCP/IP packets might be warranted. This could quickly reveal anomalies worthy of further investigation, isolation, or eradication.

Every Successful Business Needs a cTrace Packet Analyzer

As we have shown here, reading Component Traces is difficult at best. The process of collecting and analyzing these traces manually is a very tedious and time consuming task that is often difficult to justify, despite its inherent value. Companies are often forced to outsource their most complex trace analysis issues, costing both time and money that could be saved by retaining in-house control.

Ever-advancing technology makes it increasingly difficult to keep pace with the demands for performance and increasingly comprehensive diagnostic information. Even with network-based packet collecting tools such as Sniffer it is harder than ever to identify the root cause of a problem or bring hidden connectivity issues to light since technicians lack the ability to see the z/OS TCP/IP stack side of traffic flows.

A cTrace Packet Analyzer provides a comprehensive, efficient way to accelerate and simplify cTrace packet analysis. In brief, it should be easy to start (preferably automated); it should provide an abundance of usable information with a way to filter and easily digest the results; and it should create concise, accurate reports with a means to customize them to better suit individual users.

A professional cTrace Packet Analyzer restores the value of the TCP/IP component trace as an essential in-house diagnostic tool. It provides an unsurpassed utility for network technicians, dramatically reducing time-consuming, tedious analysis and making inroads into TCP/IP network problem solving. AES CleverView for cTrace Analysis was created to uniquely answer to these needs.

