16

Expert System Based Network Testing

Vlatko Lipovac
University of Dubrovnik, Croatia

1. Introduction

Networks today are not isolated entities as local-area networks (LAN) are often connected across campuses, cities, states, countries, or even continents by wide-area networks (WAN) that are just as diverse in their hardware interfaces and software protocols as LANs and may consist of multiple technologies including, too. Protocols are implemented in combinations of software, firmware, and hardware on each end of a connection. The state-of-the-art networking environment usually consists of several network operating systems and protocol stacks executing. A particular protocol stack from any manufacturer should inter-operate with the same kind of protocol stack from any other manufacturer because there are protocol standards that the manufacturers must follow. For example, the Microsoft Windows® TCP/IP stack should inter-operate with a Linux TCP/IP stack. Connections can be peer to peer, client to server, or the communications between the network components that create or facilitate connections such as routers, hubs, and bridges [1].

As converged IP networks become widespread, increasing network services demand more intelligent control over bandwidth usage and more efficient application development practices to be implemented, such as traffic shaping, quality-of-service (QoS) techniques etc. So, there is a growing need for efficient test tools and methodologies that deal with application performance degradation and faults. Network professionals need to quickly isolate and repair complex and often intermittent performance problems in their network and effectively hand over problems that are outside the network domain to the appropriate internal group or external vendor. Among the key technologies that are used for a variety of critical communication applications, we face a rapid growth of network managers' concerns, as sometimes they find their networks difficult to maintain due to high speed operation, emerging and escalating problems in real time and in a very complex environment such as: incorrect device configuration, poorly architectured networks, defective cabling or connections, hardware failures etc. On the other hand, some problems do not cause hard failures, but instead may degrade network performance and go undetected.

In particular, network management in such a diverse environment encompasses processes, methods and techniques designed to establish and maintain network integrity. In addition to its most important constituent - fault management, network management includes other activities as well, such as configuration management of overall system hardware and software components, whose parameters must be maintained and updated on regular basis.

On the other hand, performance management involves monitoring system hardware and software components' performance by various means. In addition, these tasks include monitoring network efficiency, too.
Finally, security management is gaining importance as both servers and fundamental network elements, such as bridges, routers, gateways and firewalls, need to be strictly administered in terms of authentication and authorization, network addresses delivery, as well as monitored for activities of a profile, other than expected.
Consequently, integrated network management is a continuum where multiple tools and technologies are needed for effective monitoring and control.
There are two fundamentally different approaches to network management: reactive and proactive. The reactive approach is the one most of us use most of time. In a purely reactive mode, the troubleshooter simply responds to each problem as it is reported by endeavouring to isolate the fault and restore service as quickly as possible. Without a doubt, there will always be some element of the reactive approach in the life of every network troubleshooter. Therefore, in the first phase - reactive management, IT department aims to increase network availability, where the required management features focus on determining where faults occur and instigating fast recovery.

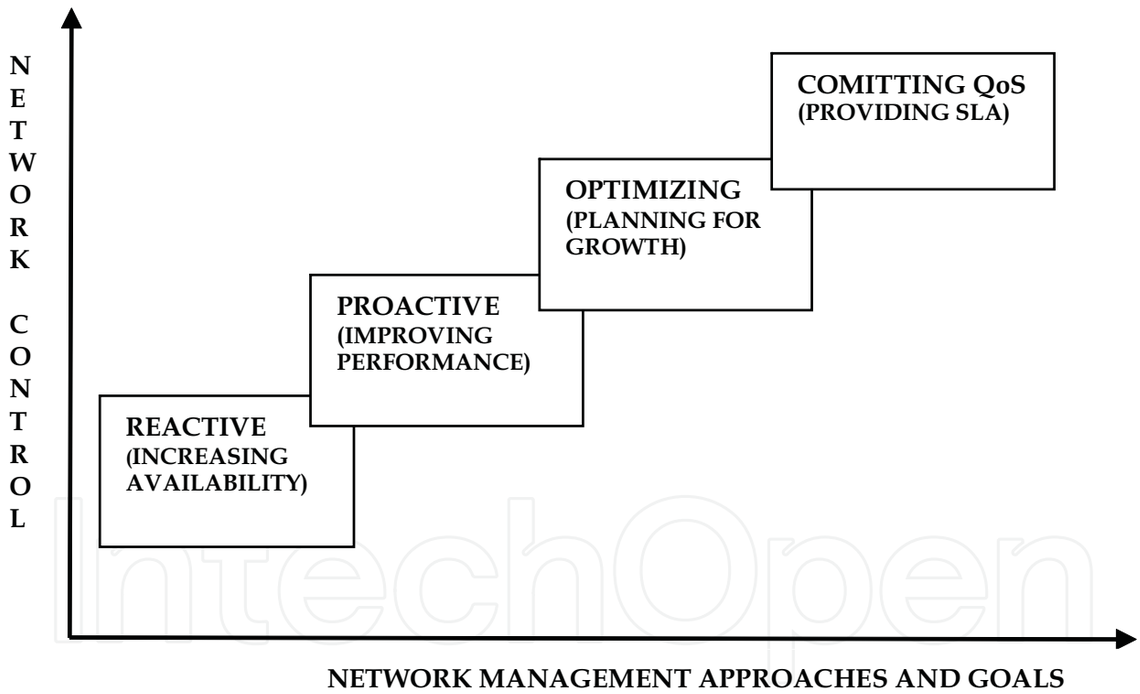


Fig. 1. Network management requirements escalation

The next phase towards increasing the network control, Fig. 1, is proactive management, which seeks to improve network performance. This requires the ability to monitor devices, systems and network traffic for performance problems, and to take control and appropriately respond to them before they affect network availability.
Optimization is the third phase that includes justifying changes to the network either to improve performance or maintain current performance, while adding new services. Trend analysis and network modeling are the key capabilities needed for optimization.

Finally, guaranteed service delivery phase involves gaining control of the criteria on which the IT organization is judged. Actually, modern network management should be primarily based on a service level agreement (SLA) that specifies precise quality characteristics for guaranteed services, stating the goals for service quality parameters that the network manager's critical responsibility is to achieve in terms of: average and minimum availability, average and maximum response time, as well as average throughput.

Nevertheless, in what follows, we will mostly be focusing troubleshooting. Analysts have determined that a single hour of network downtime for the major worldwide companies is valuated at multiple hundreds of thousands of dollars in lost revenue, and as much as about 5% of their market capitalization, while the average cost per hour of application outage across all industries, is approaching hundred thousand dollars, and is still rising. For industries such as financial services, the financial impact per hour can exceed several millions of dollars.

With this respect, the question that comes up here is whether the troubleshooting must be reactive?

Not necessarily, as the concept of *proactive troubleshooting* goes beyond the classic reactive approach, presuming that active monitoring and managing network health on an on-going basis should proceed even during the state of the network when it appears to be operating normally. By this way, the network manager is able to anticipate some problems before they occur, and is so better prepared to deal with those problems that cannot be anticipated.

Being proactive means being in control. Certainly, no one would argue against being in control. But exactly what does it take to be proactive? First of all, it takes investment of time it takes to actively monitor network health, to understand the data observed, to evaluate its significance and to store it away for future reference. Secondly, the right tools are needed, i.e. the test equipment that is capable of making the measurements necessary to thoroughly evaluate the network health. The test equipment should be able to accurately monitor network data, even during times of peak traffic load (in fact, especially then!), and intelligently and automatically analyze the acquired data.

1.1 Network management tools

There exists a wide range of appropriate test solutions for design, installation, deployment and operation of networks and services. Their scope stretches from portable troubleshooting test equipment to centralized network management platforms, where each tool plays a vital role by providing a window into a specific problem area, and is so designed for a specific purpose. However, no tool is the magic answer to all network fault management issues and does not everything a network manager typically needs to do.

Network management tools can be compared in many different dimensions. In Fig. 2, they are rated in terms of their strength in isolating and managing network faults, as well as in terms of breadth [2], [3]. With this respect, tools range from inexpensive handheld test sets aimed at physical level installation and maintenance, through built-in network diagnostic programs, portable protocol analyzers, distributed monitoring systems for multi-segment monitoring, and finally, to enterprise-wide network management systems. Many of the tools are complementary, but there is also quite a bit of overlap in capability.

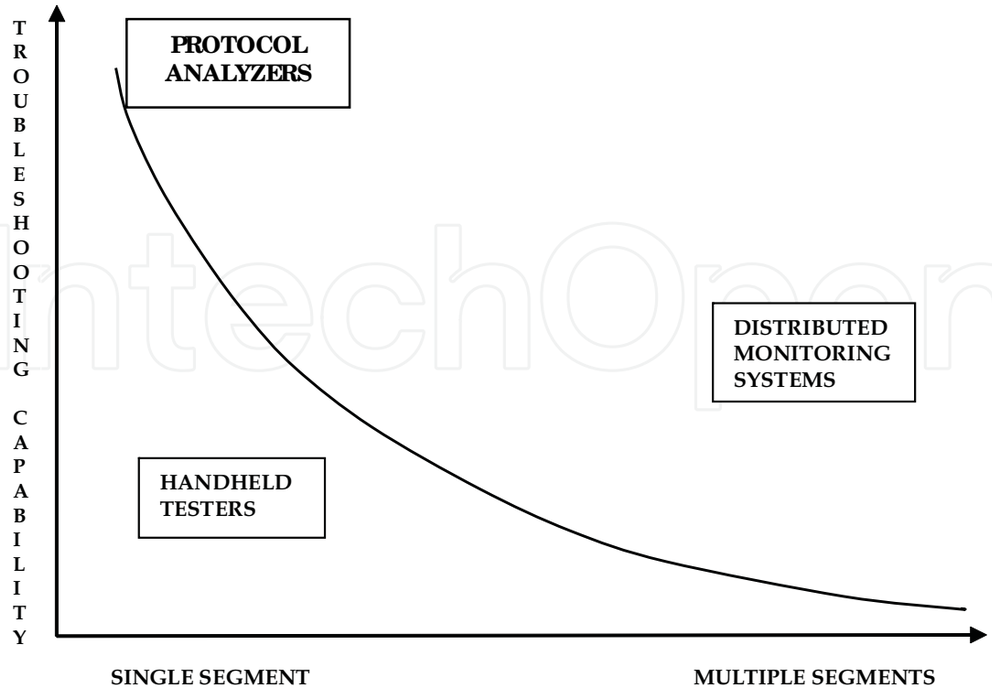


Fig. 2. Tools for fault isolation and analysis

Precise measurement of physical transmission characteristics is essential mostly for WAN testing, where tactical test instruments include interface testers and BER testers. Interface testers, also known as breakout boxes, display activity on individual signal lines, and are often used to perform quick checks on interface operating states at the first sign of trouble. They feature high-impedance inputs, so a signal under test is not affected by the testing activity and measurements can be made without interrupting network operation. In addition to simple go/no-go test panel indicator red and green LED lights, many interface testers have a built-in wiring block. This feature allows the operator to temporarily modify individual signal lines for test purposes.

Logic analyzers, oscilloscopes, or spectrum analyzers are sometimes required as well to make measurements that complement BER and interface testers and are helpful to determine the source of transmission problems. Other specialized line testing instruments, available for WAN troubleshooting, include optical time-domain reflectometers (OTDR) for fiber-optic links, and signal level meters for copper cables.

Protocol analyzers and handheld testers each view network traffic one segment at a time. Simple tools provide protocol decodes, packet filtering and basic network utilization, as well as error count statistics. More powerful tools include more extensive and higher level statistical measurements (keeping track of routing traffic, protocol distribution by TCP port number, etc...), and use expert systems technology to automatically point out problems on the particular network segment of interest.

On the other hand, distributed monitoring systems dramatically reduce mean-time-to-repair (MTTR) by eliminating unnecessary truck rolls for most network problems. These are designed to monitor multiple network segments simultaneously, using multiple data collectors – either software agents or even dedicated LAN or WAN hardware probes, generally semi-permanently installed on mission-critical or other high-valued LAN/WAN segments, to collect network performance data, typically following the format of the Remote

Monitoring (RMON) protocol [2], [3], which allows continuous monitoring of network traffic, enabling observation of decodes and keeping statistics on traffic that is present on the medium. The so acquired data are then communicated to a central network management analytical application, by means of in-band messages (including alarms when certain thresholds are exceeded), according to a certain protocol, mostly the Simple Network Management Protocol (SNMP), so that the software agents that reside on each of the managed nodes, allow the management console to see information specific to that node (e.g., the number of octets that have transited through a certain interface), or about a particular segment (e.g., utilization on a particular Ethernet bus segment), and control certain features of that device (e.g., administratively shutting down an interface).
For network troubleshooters, understanding how tool selection changes with the progress through troubleshooting process, is critical to being efficient and effective.
Strong correlation exists between a diagnostic tool being distributed or not, and whether the tool is used to *isolate* or to *analyze* network and system problems. In this sense, generally, strategic distributed monitoring tools are preferable for isolating faults, however, as compared to protocol analyzers, distributed monitoring systems (and handheld troubleshooting tools, too) typically provide only simple troubleshooting capability, while localized tactical tools - protocol analyzers usually come equipped with very detailed fault isolation capability, and are so preferable for investigating the problem cause in a local environment of interest [2], [3].

1.1.1 Protocol analysis

A simplified schematic diagram of data communications network is shown on Fig. 3, where traffic protocol data units (PDU) flow in between the user side (represented by the Data Terminal Equipment – DTE), and the network side (represented by the Data Communications Terminating Equipment - DCE). A protocol analyzer is considered as a device capable of passive monitoring of traffic and analyzing it either in real time, or in post-processing mode.

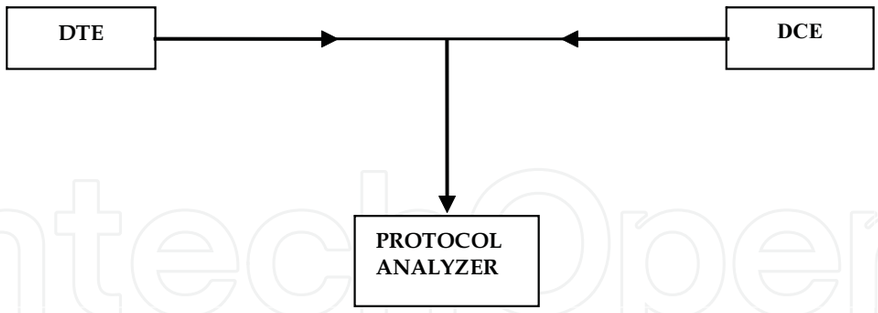


Fig. 3. Data communications network with a protocol analyzer attached in non-intrusive monitoring mode

The very essential measurement of any protocol analyzer is *decoding*, which is interpreting various PDU fields of interest, as needed e.g. for discovering and/or verification of network signalling incompatibility and interoperability problems. So, e.g. the decoding measurement of a protocol analyzer, presented on Fig. 4., displays in near real-time, the contents of frames and packets in three inter-related sections: a summary line, a detailed English description of each field, and a hex dump of the frame bytes, also including the precise timestamps of PDU (frame or packet) arrivals - crucial information that was used in the exemplar tests and analysis (characterizing congestion window) to follow in the next chapter [4].

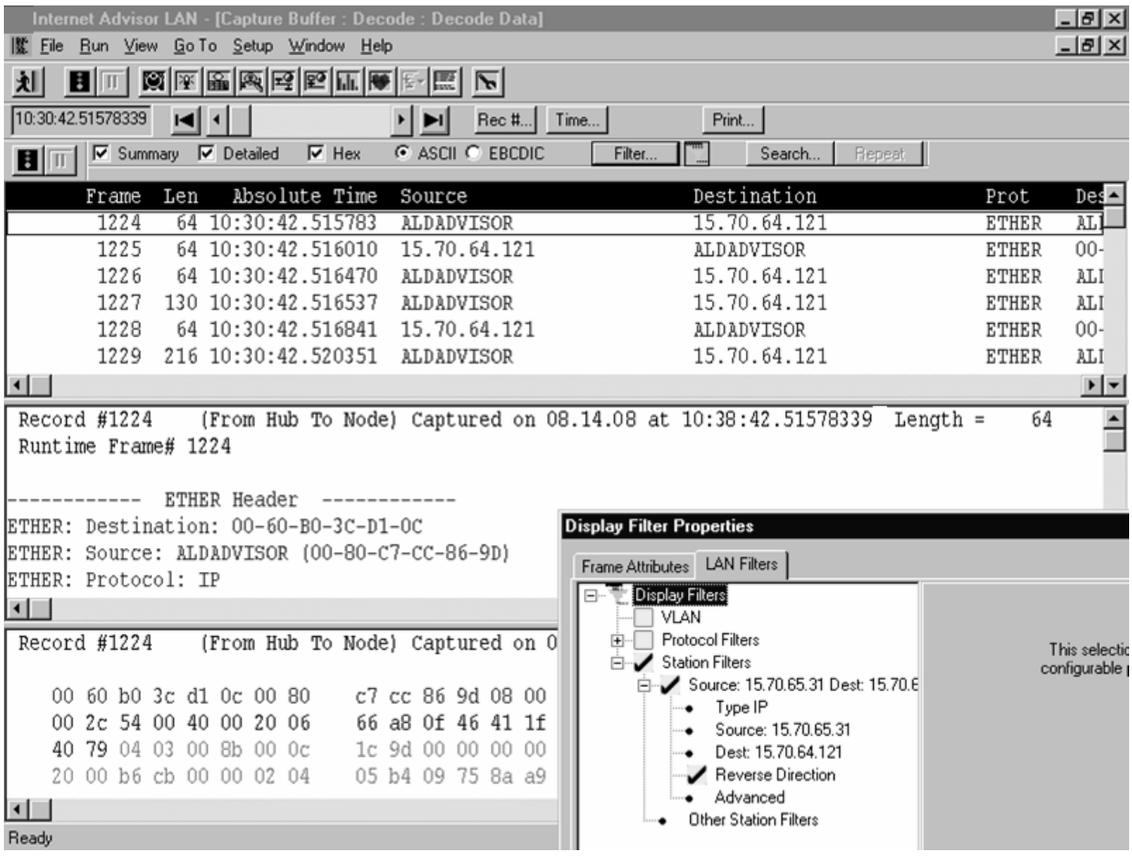


Fig. 4. Decoding of PDUs, with precise time stamping (100 ns resolution)

Specifically, the primary application of portable protocol analyzers is in-depth, detailed troubleshooting. But it would be wrong to automatically classify these powerful troubleshooting tools as appropriate only for top network protocols specialists, as state-of-the-art protocol analyzers provide powerful statistical measurements and expert systems capabilities which make these tools extremely easy to use, even for to less trained network staff.

In this sense, advanced state-of-the-art statistical analysis of traffic for a selected test station of interest often includes mutually correlated identification and characterisation of active nodes, their associated protocols and connected nodes, so providing an insight into the overall network activity of interest. So, for each active protocol stack and each active connection, line utilization and throughput (average, minimum or maximum), frame length and the number of bad frame-check-sequence (FCS) errors, will be indicated by these statistics measurements.

From the hardware platform point of view, there are several classes of portable protocol analyzers as well. However, the best type of analyzer to select depends on the size, complexity, and topology of the network involved.

Simple and inexpensive software-only applications run on standard network interface cards (NIC) and decode protocol frames, adding only rudimentary statistics measurements, while being capable of keeping up with network traffic only on low and moderately loaded networks. With limited data filtering or triggering, such products are moderately priced, and typically consume the host PC resources when running.

However, in high-speed networks, such as e.g. 1/10/100 Gbit/s LANs, higher-performance interface adapters and fast PC systems must be used in order to cope with high data volumes to be expected. Since standard NICs cannot accept all PDUs if their number exceeds a certain limit, some of them might be dropped (among them likely to be the ones most relevant for troubleshooting). In this sense, a very fast PC CPU is needed to be able to not only accept and process PDUs, but also ensure that filtering and triggering functions are performing in real time. On top of that, receive and transmit capture buffers must be deep enough, preferably with direct memory access (DMA), so to not share memory with other tasks of PC applications (among them the protocol analysis software). The NIC must have the option to switch off the local-only mode of operation (when, apart from the incoming PDUs bearing its own address, it would have seen only broadcasts), and so be able to forward all traffic it sees to the protocol analysis software.
On the other side, top high-performance protocol analyzers, Fig. 5, may also be built on a PC platform, but include special buffer memories, typically 256 Mbytes or more deep, which can be written to at very high speed, so insuring 100% data capture even under extreme traffic loads. The PDUs from such a dedicated capture buffer are processed by a RISC-based CPU, optimized for speed and accuracy, which feeds the information to protocol analysis application, running on the PC.

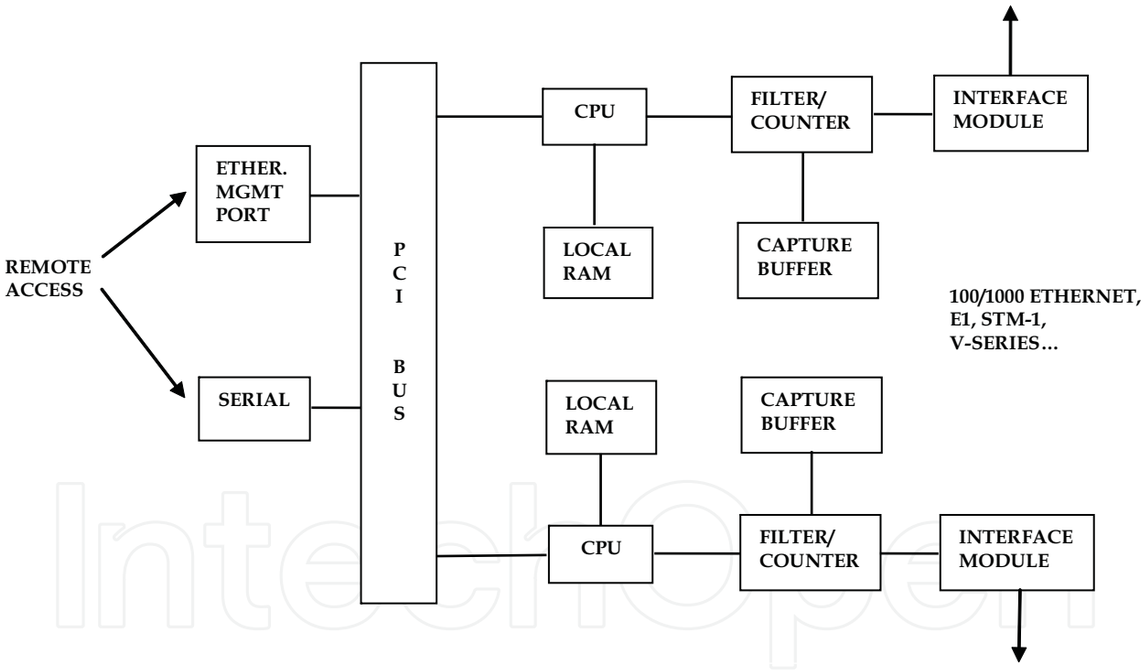


Fig. 5. Architecture of a dual-port HW protocol analyzer [4]

Such portable analyzers with dedicated hardware-based data acquisition, definitely provide much better capture performance than their software-based counterparts, as not only can they analyze and record all network traffic (time-stamped with great precision due to the high-resolution internal clock), but also generate network test traffic at wire speed (even in high-speed networks such as e.g. 10 Gb/s Ethernet). With such dedicated hardware, filtering can be accomplished in real time, regardless of filtering criteria (based on protocol, nodes and/or frame content) and instantaneous network utilization (whose peaks are most likely to coincide with eventual problems, and so are most needed to get captured and

forwarded to the analysis). In addition, some real-time trigger actions (such as e.g. event-driven stops of data acquisition) can be associated to filtering. Furthermore, hardware-based protocol analyzers usually support simultaneous multi-port measurements and so enable performance testing on multiple LAN and WAN interfaces, e.g. on both sides of network components such as routers and bridges.

2. Expert network protocol analysis

As it was already mentioned, state-of-the-art high-end protocol analyzers often contain very powerful measurement sets providing much more information than just protocol decodes. This always includes statistical analysis of traffic, and, finally, the expert analysis, where the system compares network problems that occur to information in its knowledge database, and if any error scenario is found in the database that matches the discovered situation, the system suggests possible diagnoses and troubleshooting tips, so enabling automatic fault isolation [4].

This has become more and more necessary to adequately address the diversity of potential complex network problems that definitely deserve more comprehensive approach than just using traditional network troubleshooting, which is typically based on passive network measurements by means of a classic protocol analyzer, combined with a variety of ad hoc tests. Such methodology was satisfactory when network topologies were simple and faults resulted mostly from configuration or hardware and wiring failures, i.e. when the majority of network problems were in the physical layer of the protocol stack. Nowadays, with more intelligent network devices (e.g. integrated layer 2 switching and layer 3 routing), application/server load balancing (i.e. layer 4 - 7 switching and routing), more sophisticated security policies and devices, as well as QoS technologies, most network problems have crept up the stack. Consequently, unlike before, a rising percentage of network performance problems, faced by network managers, are attributed to higher OSI layers, namely 3, 4 through 7, rather than hard failures. These can include network software bugs, too many users trying to use the network at once and/or soak up the available network bandwidth, interoperability problems between protocol stacks because of different interpretations and implementations of standards, breached network security or software and hardware updates with unexpected results etc. Moreover, as deploying state-of-the-art complex, multi-services and multi-technology high-speed networks, including data, voice and streaming media applications, has become reality, delivering high-availability communication infrastructures for mission critical applications, and contracted QoS, as well as maintaining fast growing of sophisticated networks, imply that network downtime is not an affordable option at all. On top of that, dramatically rising network problems complexity also implies longer *Mean-Time–To-Repair* (MTTR), even without taking into account that quite often network managers rely on limited skill personnel.

Specifically, even a protocol analyzer that is equipped with the best data acquisition hardware and application-level decodes, as well as advanced statistical analysis (that identifies how busy is the network, who is connected to it and is using most network bandwidth, which protocols are the most active stations using, when they are using it, for what, and whether the network is reaching its capacity, etc.), in many instances, will not itself timely isolate complex problems on the network and diagnose who is causing them, if still the fault management process is mostly manual and so very time-consuming, requiring a great deal of expertise in many aspects of network technology.

This in turn means that, in order to keep up with the next-generation-network (NGN) challenges, troubleshooting methods have to change, as well to better address the rising need for more sophisticated test tools that will make the process more efficient by providing automated means for continuous higher-level identification of problems, with less human intervention, so making decisions on how to best manage the network, justified and so easier.
With this respect, state-of-the art protocol analysis often incorporates some sort of an expert system that offers a beneficial solution to these problems by continuous monitoring a network for performance degradation and faults in all 7 OSI layers, logging the results and then looking up its knowledge database, searching for eventual similarities with the currently identified network problems. Thus, the expert system capability of a protocol analyzer essentially not only takes advantage of but goes well beyond passive protocol following and statistical performance measurements, thus making fault diagnosis much more comprehensive. The intelligent protocol decodes automatically report on errors or deviations from the expected protocol, so reducing thousands of captured data frames to a short list of significant network events, and interpreting the significance of these events. This way, appropriately reported, such expert-analysis-isolated network abnormalities and inefficiencies significantly reduce the scope of potential causes of the problem (if not self-sufficiently pinpoint what it most likely might be), suggesting possible solutions that the network manager can consider to figure out what is wrong from the visible symptoms, so greatly speeding up the troubleshooting process. In other words, hand-in-hand with the proactive troubleshooting process is another methodology called *intelligent analysis*, which refers to the use of state-of-the-art data reduction tools available on today's test equipment, which facilitate rapid fault isolation, as a way to avoid network troubleshooting in (purely) reactive chaos.
Better yet, this way, network managers can even predict the possibility of network failures and take actions to avoid the conditions that may lead to problems.

2.1 Expert protocol analysis system basic components

A typical rule-based expert system consists of five components: knowledge base, inference engine, blackboard, user interface, and explanation facility, Fig. 6.

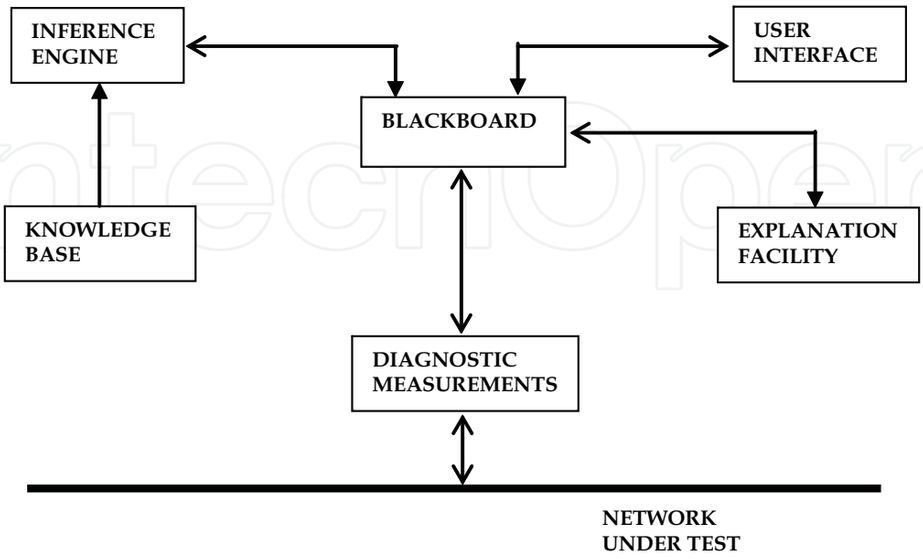


Fig. 6. Typical rule-based network expert system components

The knowledge base is a collection of data that contains the domain-specific knowledge about the problems being solved. The inference engine performs the reasoning function. It is the component of the inference engine that controls the expert system by selecting the rules from the knowledge base to access, execute and decide when a solution has been found. After performing measurements and observing the network for significant events, pending measurement results that satisfy the rules' preconditions are posted to the blackboard. The blackboard is a communication facility that serves as a clearinghouse for all information in the system, while the user interface allows the user to input information, control the reasoning process and display results. The explanation facility interprets the results by describing the conclusions that were drawn, explaining the reasoning process used, and suggesting corrective action.

An expert system used for network troubleshooting must have access to diagnostic functions to actively confirm the existence of faults and to gather information from other devices and network management systems on the network. It must generate alarms and log all pertinent information in a data base. Automated operation must be available so that human intervention is not required, and audit trails should be provided so that a network manager can later track problems.

An expert system must also be able to proactively obtain information about the state of the network to prove (or disprove) hypothesized problems. This is performed by the rules requesting information (via the inference engine) from the blackboard. The results of the measurements are posted to the blackboard to allow the inference engine to continue and eventually prove (or disprove) the hypothesized problem.

Often, real-time demands of troubleshooting a network exceed the performance capabilities of a conventional rule-based expert system. However, intelligent measurements can greatly improve the performance of a rule-based expert system. Measurements are considered to be intelligent if they actually interpret the information on the network, instead of simply reporting low level events such as frame arrivals. An example of an intelligent measurement would be one that monitors the network and provides a high-level commentary on significant network events and conversations between nodes by following the state of network transactions. It would indicate if a connection was established properly, ensure that the traffic level between nodes did not exceed a maximum limit, and identify new mappings between physical layer addresses and network addresses. The process of managing networks includes fault detection and isolation. Network faults refer to problems in areas such as physical media, traffic and routing, connection establishment, configuration and performance.

In what follows it will be briefly described how an expert system, embedded in a protocol analyzer, addresses the areas of fault detection and isolation, specifically in solving common faults in a complex network environment.

2.2 Network baselining and benchmarking

Understanding how the network under test operates is crucial in managing it proactively, so without it, a network manager will not have the information needed to make sound decisions concerning how his network is managed.

Does he have misconfigured servers and nodes that are sending extra data onto the network? Is the network overloaded? Is it time to upgrade hardware or software? Has that recent department relocation had an adverse affect on the network? How much growth has occurred on the network over the past year? Can it sustain that growth level for another year?

Two key processes need to be implemented to proactively manage the network: first, and most important, the network manager must *baseline* the network, so to get understanding

who is using it and how it is being used. Essentially, a baseline is a set of statistical measurements made over a period of time, which characterizes network performance, Fig. 7. It is a snapshot of the characteristics of the network of interest.

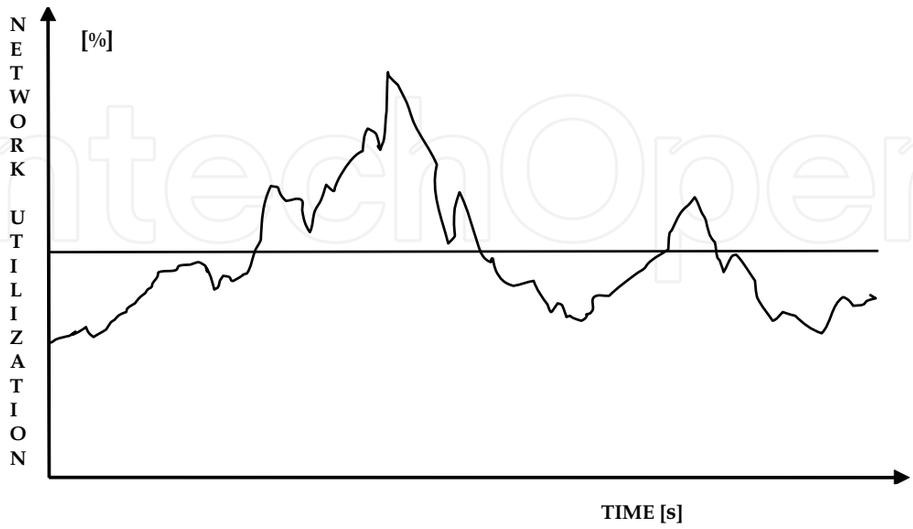


Fig. 7. An example of network baseline

Measurement results from recorded baselines describe normal operating conditions of the network, and so can provide points of reference - thresholds for future advanced statistical analysis and expert measurements, thus enabling discovery of eventual departures of multiple measurements results from their belonging thresholds, by reporting them as significant events (e.g. for TCP/IP or XoIP protocols), should anything go wrong sometime later. With properly set thresholds, e.g. such as the ones in Fig. 8, significant changes will be neither missed, nor unnecessarily interpreted as routine events.

Some of the so detected high-level ordered significant protocol events may indicate normal and desirable activity, while others might indicate the presence of potentially serious problems that should be present only in very rare instances. Following this classification are the "normal", "warning", and "alert" events, enabled in the configuration window of the above example, sorted by the order of their severity in indication that the identified problems could lead to network performance degradation or network failure [4].

By this way, baselining uncovers any network problems or inefficiencies so that they can be fixed before their major affect on users, which also enables better planning for growth. By looking at successive baselines, taken regularly over a long period of time, one will be able to observe trends and, based on them, plan for future, in terms of capacity growth and investments. Moreover, benchmarking applications and components before they are installed on the network provides the information needed to understand and predict their effect.

Thus, using the tactics of baselining to first understand the normal network operation and, when problems arise, perform another baseline and compare the results, problems can be quickly identified and/or inefficiencies in network operation exposed. This provides immediate opportunities for improving network performance by observing trends and recognizing changes, and so being able to anticipate and resolve problems before they become apparent to network users.

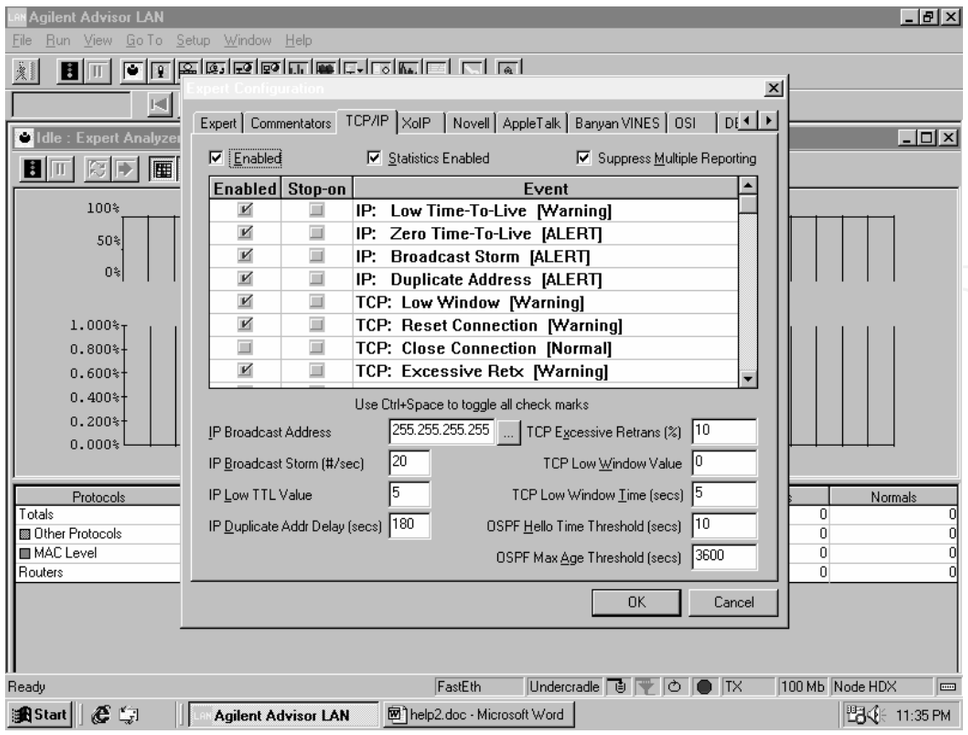


Fig. 8. Example of selected TCP/IP significant events for the related expert protocol analyzer measurements, and setup of baseline thresholds, to match particular network conditions

There are three main steps in performing a network baseline: collecting the data, creating the report and interpreting the results. First, the data must be collected either using a protocol analyzer, or a distributed network monitoring system agent, connected directly to the network segment of interest, such as the backbone and server ones, or the segments interconnecting separate networks (WAN interconnections first). The data should be collected for a fixed period of time, at similar time periods and at regular intervals, especially before and after large network adds, moves, or changes.

Before beginning the data collection process, one will first have to choose a sample period, which is the total period of time over which baseline measurements are made. The sample interval is the period of time over which each individual statistics is sampled and averaged, i.e. it is the amount of time between data points in the baseline data - the time resolution used to collect the data samples, Fig. 9.

As the sample interval gets larger, it will be less possible to resolve rapid changes in measured characteristics, as they will be averaged out. So, small sample intervals and small measurement periods should be used when fine resolution is required, which is usually appropriate for fault isolation or to obtain an instantaneous characterization of network health. On contrary, longer sample intervals and longer overall measurement periods are recommended when baselining for long-term trends, or gaining an overall picture of network health.

Typically, most appropriate sample intervals are e.g. 1 second samples for periods up to 2 hours, 1 minute samples for periods up to 24 hours, or 10 minutes to 1 hour samples, for periods of two or more days.

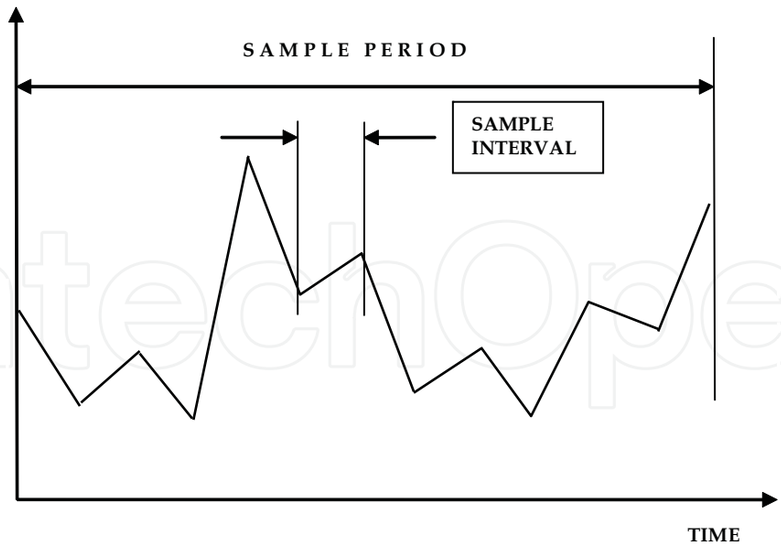


Fig. 9. Baseline sample interval and sample period

Both portable protocol analyzers and distributed monitoring systems provide the information necessary to baseline and benchmark the network, but, as it was already pointed out, each of them has its unique capabilities that can help troubleshoot problems or monitor network performance, respectively. Whatever devices are used as data collectors, enough data must be provided, with enough accuracy to provide the real insight into the network operational characteristics. Each network segment of interest should be baselined, as e.g. a computer-aided engineering (CAE) segment will have quite different characteristics than a segment running just business applications.

Collecting the data is important; however, if they are not presented in a clear and meaningful way, it will be very difficult to interpret them, as finally the network manager needs to look for abnormalities (such as e.g. high levels of network utilization, low average data packet size, high level of errored frames, or a file transfer protocol (FTP) with average data size of 100 bytes –indicating that the client or server could be configured incorrectly or overloaded etc.), trying to learn what is normal for his network over time, and comparing successive baselines to question any significant changes in traffic patterns or error levels. The baseline reports can also be used to set thresholds to be used as input to a rule-based automatic expert system that will review the baseline instead of a human network troubleshooter, and look for abnormal symptoms, to identify and question unusual traffic patterns for multiple protocol suites of interest. This will not only help in understanding how the network operates, but also to predict future changes in the network behaviour, before they actually occur (with potentially troublesome consequences).

2.3 Practical expert protocol analysis

The expert analysis must be executed on a truly multitasking machine, able to simultaneously and automatically run several measurements, comparing the measured values with their corresponding thresholds and so "feeding" the decision algorithm with input data. With such an arrangement in protocol analysis, PDUs are decoded nearly real-time, where the only reason for not fully real-time decoding is that other simultaneous processes can slow it down a

bit. Intelligent expert system-based protocol decodes automatically follow each conversation, and report on errors or deviations from the expected protocol.
However, this usually comes integrated with a number of other powerful intelligent tools – mostly high-level protocol and node statistical analysis, which provide automatic node and protocol events discovery, and even complete network health audit. These intelligent analysis tools do much of the problem analysis work automatically, by separating the significant few events from thousands and millions of PDUs passing through the analyzer every second, where examining the details in individual PDUs for each protocol stack running only makes sense after real-time narrowing the focus (by the intelligent tools) just to significant events, such as connection resets, router misconfigurations, too slow file transfers, inefficient window sizes, and a number of other problems that might occur. Created this way, a short list of significant network events with their belonging interpretations and classifications, could suggest most likely network faults, so enabling quick high-level identifying network problems, i.e. the trade-in between the time to identify a problem, and the (so extended) time to solve it, thus greatly increasing the productivity by automating this process.
Most manufacturers call this capability an expert system or expert protocol commentator [2], [3], [4].
An illustrating example of a typical expert analysis top-level result is presented on Fig. 10, where too many retransmissions at TCP layer have been reported, slowing down the network.

Fig. 10. An example of expert analysis based detection, isolation and classification of excessive TCP transmissions

As it can be seen from the display, the related event is classified as "warning", showing the node and connection information in one view, properly time-stamped, as well as possible causes (most likely noisy lines and/or inadequate IP *Time-To-Live* (TTL) setting). The alternative might be searching through decodes of thousands of captured frames to

eventually figure it out. This hypothesis can be further investigated and verified through examining the frames with bad cyclic redundancy check (FCS), as well as through active out-of-service bit-error-ratio (BER) tests. (In general, some network faults just cannot be isolated without such stimulus/response testing. For example, observing Ethernet frames with the same IP address and different MAC addresses might indicate a duplicate IP address problem - but it might also be just a consequence of a complex router topology. However, ARPing the stations for their addresses will resolve the dilemma in seconds).

Other common examples of expert troubleshooter findings include e.g. router misconfigurations, slow file transfers, *inefficient window sizes* (that was used in congestion window analysis example to follow), TCP connection resets, protocol anomalies and mis-sequencing, inefficiently configured subnets (so enabling too much cross-subnet traffic and a router busier than it is necessary), utilization too high, too many broadcasts/multicasts or too much management traffic (both using considerable bandwidth), one or more computers transmitting errors, and many more.

On top of the advanced statistical analysis of active connections, stations and nodes involved, as well as expert classifications of significant events, often a sort of network "health" figure is estimated, based on the number of identified more or less severe events, whom the appropriate weighting factors are assigned to subtract the adequate amount (per each such identified event) from the value of 100%, as presented in Fig. 11, where from the top level display of the network health, more detailed investigations can be opened by drilling down into the related embedded expert or statistical analyses [4].

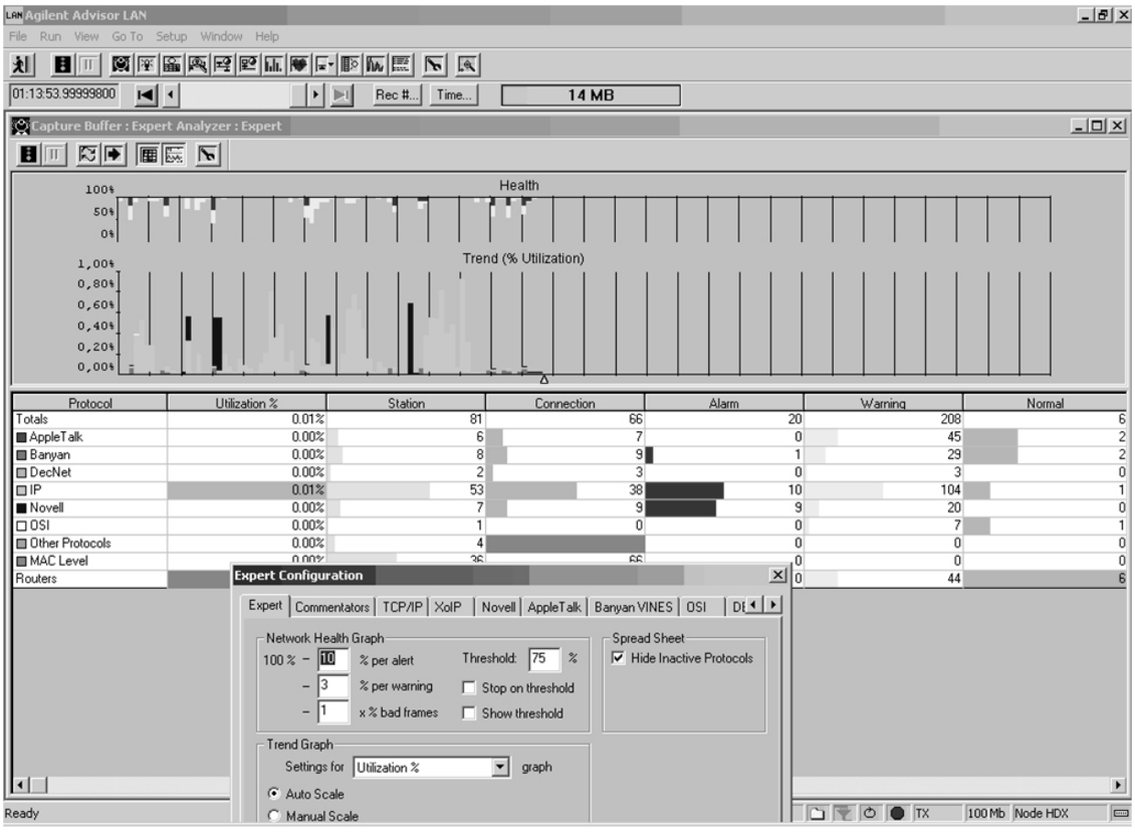


Fig. 11. Network health

3. An example of testing TCP traffic congestion by expert protocol analysis and statistical modelling

In what follows, an exemplar solution for expert-system-based distributed protocol analysis of TCP congestion window process in a major network with live transaction-intensive traffic (specifically, electronic financial transactions data transfer), during stationary time intervals, is presented [5], based on estimating actual congestion window size from measured data that mainly included decoding with precise time-stamps (100 ns resolution locally and 1 μs with GPS clock distribution), and expert-system findings, resulting from appropriate processing of network data, accordingly filtered prior to arriving to the hardware-based capture buffer.
In addition, a statistical analysis model is presented for evaluation whether the observed protocol data belonged to the specific (in this case, normal) cumulative distribution function, or whether two data sets exhibit the same statistical distribution - the condition-sine-qua-non for a stable interval with regard to TCP. Having identified such stationary intervals, the measured-data-based congestion window values were found to fit very well (with satisfactory statistical significance) to the truncated normal distribution. Finally, an appropriate model for estimation of relevant parameters of the congestion window distribution - its mean value and the variance, was developed and applied.
Transport Control Protocol (TCP) is a connection-oriented and so reliable protocol that much of Internet traffic uses at transport layer (the rest belongs to the connectionless User Datagram Protocol (UDP)). As a (sliding) window-based protocol, it controls sending rate at end-points, together with queuing mechanisms provided by routers [1].
It is the imperative to predict the performance of connections, so with this regard, the means for testing the congestion window process through experimental approach, is proposed here, as real Internet traffic was measured, the collected data processed (the so far elaborated way), and the additional statistical methods selected for case-specific analysis.

3.1 Flow control and managing congestion in TCP/IP networks

As it can be seen in Fig. 12, each TCP traffic PDU – segment is divided into the header and the data.

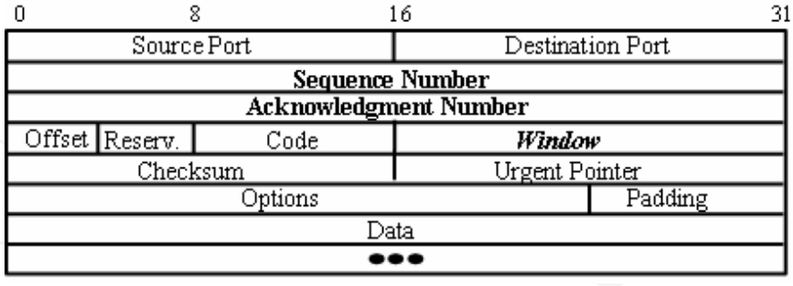


Fig. 12. TCP segment format

The TCP window is sometimes referred to as the "TCP sliding window" [1]. This field tells the receiver how many bytes the transmitting host can receive without acknowledgement. Thus, the sliding window size in TCP can be adjusted in real time, so it is a primary flow control mechanism, allowing more sender data to be "in flight", so that, this way, the sender gets ahead of the receiver (though not too far). Actually, the so advertised window informs sender of receiver's buffer space.
In original TCP design, this was the only protocol mechanism controlling sender's rate. However, this simple flow control mechanism keeps a (faster) sender from flooding with

traffic a (slower) receiver, while congestion control prevents a number of senders from overloading the *network*, adjusting to bandwidth variations, as well as sharing it among various data streams.

In real life, congestion is unavoidable; when two packets arrive at the same time, the node can only transmit one of them, and either buffer or drop the other. Specifically, if too many packets arrive within a short period of time, the buffer may eventually overflow, resulting with performance drop due to undelivered packets, packets consuming resources that are dropped somewhere else in the network downstream, spurious retransmissions of packets still "in flight" (leading to even more load)…

In mid-1980s, the Internet faced serious performance problems, until Jacobson/Karels devised TCP congestion control [1]. Generally, avoiding drops of too many packets and the so-called network congestion collapse, was based on: pre-arranging bandwidth allocations (with drawback of requiring negotiation before sending packets and potentially low utilization), differential pricing (i.e. not dropping packets for the best bidders), as well as dynamic adjusting of transmission rate that is increased when testing of congestion reflects its significant value, and is decreased otherwise (where drawbacks are: suboptimality and complex implementation) [8].

With this regard, the term "congestion window" denotes the maximum number of unacknowledged bytes that each source can have "in flight". This implies that, in other to conduct congestion control, each source must determine the available network capacity, so to know how many packets it can leave "in flight". Congestion-control equivalent of the receiver window principle should presume sending at the rate of the slowest component, in order to adapt the window by choosing its (maximal) size as the minimal out of the two values: the actual congestion window and the receiver window. So, upon detecting congestion, the congestion window must be (fast) decreased, as well as increased should no congestion was detected.

Detecting congestion by a TCP sender can be accomplished in a number of ways. For example, an indication can be if Internet Control Message Protocol (ICMP) *Source Quench* messages are detected on the network (either through protocol analyzer decoding, or expert analysis). However, this is not particularly reliable, because during times of overload, the signal itself could be dropped. Increased packet delays or losses can be another indicator, but also not so straightforward due to considerable non-congestive delay variations and losses (checksum errors) that can be expected in the network.

Anyway, no matter how congestion is detected, managing it must start from the fact that the consequences of over-sized window (packets dropped and retransmitted) are much worse than having an under-sized window (somewhat lower throughput). Therefore, upon success with last window of data, the TCP sender load should increase linearly, and decrease multiplicatively, upon packet loss [8]. This becomes a necessary condition for stable state of TCP [9], [10], [11].

Particular schemes for managing congestion window are out of scope of this paper and will therefore not be further explored here, as our experimental investigations and analysis focused just the estimation of statistical distribution of the stationary congestion window size.

The available test methods for studying communications networks range from mathematical modelling, through simulation (and/or emulation) to real-life measurements. We based this research on measuring the relevant parameters of test traffic by specialized hardware and analyzing the measurements' results by expert analysis and statistical modelling.

3.2 Architecture of the test system

A combined hybrid centralized (but) distributed expert analysis testing and troubleshooting solution, based on central server application, which controls and integrates expert protocol analysis, and distributed SNMP/RMON monitoring and analysis system, is the high-end solution in network management. Such an integrated solution consists of multiple expert protocol analyzers, RMON and other test agents, providing the means to solve complex problems, still spanning several network links and technologies and so enabling fastest progression from detection, identification and isolation of problems with availability, routing, QoS, etc., through network-wide view by RMON/MIB data, to resolution of problems by drilling-down into advanced statistics and expert analysis executed on a targeted protocol analyzer, where and when it comes out necessary (to troubleshoot).

The network under test consisted of LAN (Ethernet 100 Mbit/s to 1 Gbit/s) and WAN (Frame Relay) infrastructure of a major bank in/between their offices in three cities was used as a system under test. The network included dedicated workstations, residing at different locations around the network and exchanging test traffic, as well as of six distributed test system devices: hardware-based distributed protocol analyzers (DPA), from Agilent Technologies, aimed for protocol data acquisition and analysis. These were dispersed throughout various network segments of interest, either as network resident or temporarily installed for network performance testing. Their interwork was orchestrated by the central application server (running the Network Troubleshooting Center software of the same vendor), which controlled and integrated the distributed protocol data acquisition modules and their expert protocol analyses, and was accessible via two remote clients – consoles [5], Fig. 13.

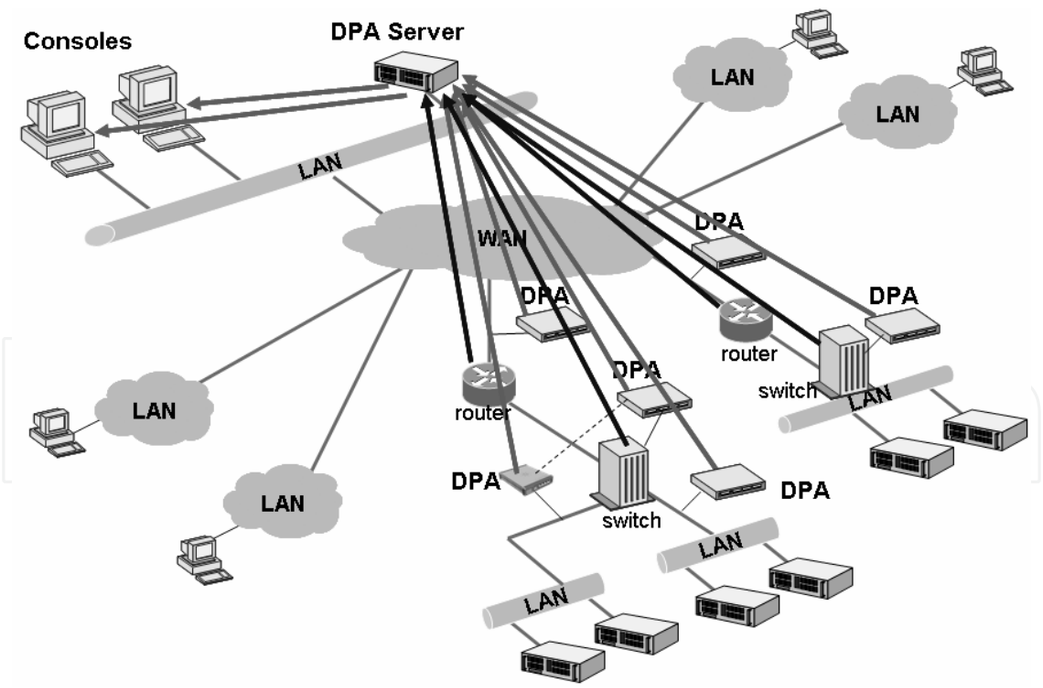


Fig. 13. Test scenario with distributed protocol analysis system spanning distant networks

By examining the precisely time-stamped TCP traffic PDUs – segments, using the tools of the non-intrusively attached DPAs, we were able to characterize the network, as well as its endpoints.

We performed many measurements throughout the day work time and in various environments: LAN – LAN and LAN-WAN-LAN. For each packet sent or received, DPAs registered its timestamp, sequence number and size. Multiple DPAs, with their 1Gb/s acquisition system, supporting real-time network data capture (to capture buffer memory) and filtering, were combined into time-synchronized multi-port tests, still using the same software features as with stand-alone protocol analyzer, such as e.g. decoding, statistical analysis and expert analysis [4]. Time synchronization among DPAs was achieved either via the "EtherSync" interfaces (where DPAs were daisy-chained, which allowed them to be synchronized to each other within ±100ns), or by means of the external GPS sources, providing the synchronization accuracy of ±1μs. (For the reference, the IETF's Network Time Protocol (NTP) could provide the synchronization accuracy of ±20ms or better, depending on the proximity of Network Time Servers (NTS)).
With this regard, the scenarios deployed included: multiple daisy-chained DPAs connected to a PC or a protocol analyzer either directly, or via a LAN, or multiple DPAs in a network with GPS, rather than NTP time synchronization.
The protocol analyzers used were equipped with both LAN and WAN interfaces, which provided physical connections and high-speed data acquisition hardware to get every frame on the network into any particular protocol analyzer. The packet slicing option was selected on capture buffer of the each interface, to enable the protocol analyzers to capture only the first part (e.g. fist 100 Bytes) of each packet, containing just the relevant header information, so that more packets for a given buffer size (of up to 256 Mbytes), could be stored. Mostly the *full buffer* option was used, where collecting data continued until the buffer was filled, when it was finally stopped.
The built-in capture filters were enabled, to control which frames were allowed to enter the capture buffer, and so to focus the analyzer (or just to save space in the capture buffer). As these filters are implemented in hardware, they were also used to trigger an action, such as halt or start collecting data on a matched frame, as well as either include or exclude matched frames from logging into the buffer, and later on to the hard disk of the analyzer's PC platform. Among a number of different available filter criteria, protocols (TCP, IP, etc.), specific fields (such as e.g. window size), frame attributes (such as erroneous or good frames etc.), and, in some instances, frame data bytes (the first 127 bytes) were used.
Associated to capture filters are statistics counters that provide counts of frames, packets and other events matching the selected filter criteria. These were set up and used for getting the precise statistics – histograms of the traffic events that were investigated.
With regard to specific measurements and data processing done, certainly the most rudimentary one was decoding from which the very essential information used for characterizing congestion window are precise timestamps of PDU arrivals, Fig. 4.
On top of data processing in each DPA, the appropriate postprocessing software (Agilent's Report Center) was used to accomplish multi-segment network baselining and benchmarking, with time correlation of data across the segments of interest. Using these multitasking measurement features enabled analyzing the raw data in different ways concurrently, such as e.g. to get correlated statistics between protocols, nodes (that use these protocols) and connections of each such end-station [4].
In order to estimate the sender's congestion window size from the collected data, it was necessary to identify (by filtering with appropriate criteria) the packets that have been sent from the sender, but have not yet arrived at the receiver, count them (by stats counters) and present as a function of time. The already presented features of the experimental system enabled fulfilling this task with great precision and accuracy. A simple application program – a counter - was used to add 1 to the actual congestion window size for each outgoing

packet, at the time it was leaving the sender, and subtract 1 when/if that packet arrived at the receiver. With the exception of lost packets (that we can trace by various means, the easiest one by the expert analyzer, Fig. 10), which were excluded from the calculation, this accumulated sum well approximated the actual congestion window size almost in real time.

3.3 Statistical analysis model

In statistics, the Kolmogorov–Smirnov (K-S) test is used to find out if an empirical cdf of interest, based on finite samples, differs from a hypothesized continuous distribution function specified by the null hypothesis [12]. The very reason for wide use of the K-S test statistic is that it does not depend on the distribution function being tested, and also that it does not depend on adequate sample size (as e.g. the chi-square goodness-of-fit does). The higher the extent to which this test implies that the set up hypothesis has (or has not) been nullified - the significance level α (of the difference between the hypothesized values and the sample-based ones), obviously, the less likely it is that the investigated event could have been produced only by chance. So, in fact, the significance level is the probability that the null hypothesis could be wrongly rejected, when actually it should be accepted [12]. (Such a decision is commonly referred to as "false positive"). Among the popular levels of significance: 5%, 1% and 0.1%, in our model, we adopted the mid value.
The significance of a result is frequently expressed as its p-value, in such a way that smaller p-value reflects more significant result. So, when p-value, resulting from such a test, is smaller than the α-level, the null hypothesis is rejected and informally speaking, the results are classified as “statistically significant”, where the lower significance level implies the stronger evidence.

3.3.1 Testing conformance to normal distribution

A sample Kolmogorov-Smirnov test [12] enables testing of a hypothesis that a certain distribution $F_{n\xi}(x)$ of a random variable ξ conforms to the given continuous cdf $F_{0\xi}(x)$.

$$H_0 : F_\xi(x) = P(\xi < x) = F_{0\xi}(x) \tag{1}$$

The empirical cdf $F_{n\xi}(x)$ is derived from the independent samples $(\xi_1, \xi_2, ..., \xi_n)$. The Kolmogorov-Smirnov statistics for a given $F_{0\xi}(x)$ is:

$$D_n = \sup_x \left| \hat{F}_{n\xi}(x) - F_{0\xi}(x) \right| \tag{2}$$

As it follows from the theorem of Glivenko-Cantelli [12], if the observed sample comes from the $F_{0\xi}(x)$ distribution, then D_n converges to 0. Furthermore, as $F_{0\xi}(x)$ is continuous, the rate of convergence of $\sqrt{n}D_n$ is determined by the Kolmogorov limit distribution theorem, stating:

$$\lim_{n\to\infty} \Pr(\sqrt{n}D_n < y) = K_\eta(y), \quad 0 < y < \infty \tag{3}$$

where $K_\eta(y)$ is the Kolmogorov cdf (that does not depend on $F_{0\xi}(x)$, as pointed out above).
Moreover, if the significance level of α is pre-assigned, then the tested null-hypothesis is to be rejected at the level α if:

$$\sqrt{n}D_n > y_\alpha \tag{4}$$

where the cut-off y_a is found by equalizing the Kolmogorov cdf $K_\eta(y)$ and 1-α:

$$\Pr(\sqrt{n}D_n \le y_\alpha) = K_\eta(y_\alpha) = 1-\alpha \;\Rightarrow\; y_\alpha = K^{-1}{}_\eta(1-\alpha) \tag{5}$$

Otherwise the null-hypothesis should be accepted at the significance level of α.
Actually, the significance is mostly tested by calculating the (*two-tail* [12]) p-value (which represents the probability of obtaining the test statistic values equal to or greater than the actual ones), by using the theoretical $K_\eta(y)$ cdf of the test statistic to find the area under the curve (for continuous variables) in the direction of the alternative (with respect to H_0) hypothesis, i.e. by means of a look-up table or integral calculus, while in the case of discrete variables, simply by summing the probabilities of events occurring in accordance with the alternative hypothesis at and beyond the observed test statistic value. So, if it comes out that:

$$p = 1 - K_\eta(\sqrt{n}D_n) < \alpha \tag{6}$$

then the null hypothesis is again to be rejected, at the presumed significance level α, otherwise (if the p-value is greater than the threshold α), the null hypothesis is not to be rejected and the tested difference is not statistically significant.

3.3.2 Identifying stationary intervals

While the main applications of the one-sample K-S test are testing goodness of fit with normal and uniform distributions, the two-sample K-S test is widely used for nonparametric comparing of two samples, since it is sensitive to differences in both location and shape of the empirical cdfs of two samples, so it is the most important theoretical tool for detecting change-points.
Let us now consider the test for the series $\xi_1, \xi_2, ..., \xi_m$ of the first sample, and $\eta_1, \eta_2, ..., \eta_n$ of the second, where the two series are independent. Furthermore, let $\hat{F}_{m\xi}(x)$ and $\hat{G}_{n\eta}(y)$ be the corresponding empirical cdfs. Then the K-S statistics is:

$$D_{m,n} = \sup_x \left| \hat{F}_{m\xi}(x) - \hat{G}_{n\eta}(y) \right| \tag{7}$$

The limit distribution theorem states that:

$$\lim_{m,n \to \infty} P\left(\sqrt{\frac{mn}{m+n}} D_{m,n} < z \right) = K_\zeta(z), \quad 0 < z < \infty \tag{8}$$

where again $K_\zeta(z)$ is the Kolmogorov cdf.

3.3.3 Estimation of the (normal) distribution parameters

Let us consider a normally distributed random variable $\xi \in N(m, \sigma^2)$, where:

$$p_\xi(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-m)^2}{2\sigma^2}} \tag{9}$$

Its cdf $\Phi_\xi(x)$ can be expressed as the standard normal cdf $\Phi(x)$ [12] of the ξ-related zero-mean normal random variable, normalized to its standard deviation σ:

$$\Phi_{\xi}(x)=\Pr(\xi\le x)=\int_{-\infty}^{x}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(u-m)^2}{2\sigma^2}}du=\Phi\left(\frac{x-m}{\sigma}\right)=\int_{-\infty}^{\frac{x-m}{\sigma}}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{v^2}{2}}dv \tag{10}$$

Normal cdf has no lower limit, however, since the congestion window can never be negative, here we must consider a truncated normal cdf. In practice, when the congestion window process gets in its stationary state, the lower limit is hardly 0. Therefore, for the reasons of generality, here we consider a truncated normal cdf with lower limit *l*, where $l\ge 0$.
Now we estimate the parameters *m*, *σ* and *l*, starting from:

$$\Pr(\xi>l)=1-\Phi\left(\frac{l-m}{\sigma}\right)=1-\int_{-\infty}^{\frac{l-m}{\sigma}}\frac{1}{\sqrt{2\pi}}e^{-\frac{v^2}{2}}dv=Q\left(\frac{l-m}{\sigma}\right) \tag{11}$$

where: $Q\left(\frac{l-m}{\sigma}\right)$ is the Gaussian *tail* function [12].

The conditional expected value of *ξ*, just on the segment $(l,+\infty)$ is:

$$E(\xi/\xi>l)=\int_{l}^{+\infty}u\cdot\frac{1}{\sqrt{2\pi}\sigma\cdot Q\left(\frac{l-m}{\sigma}\right)}e^{-\frac{(u-m)^2}{2\sigma^2}}du \tag{12}$$

By substituting: $\frac{u-m}{\sigma}=v,\ du=\sigma\cdot dv$ into (12.), we obtain:

$$\begin{aligned}E(\xi/\xi>l)&=\frac{1}{Q\left(\frac{l-m}{\sigma}\right)}\int_{\frac{l-m}{\sigma}}^{+\infty}(\sigma\cdot v+m)\frac{1}{\sqrt{2\pi}}e^{-\frac{v^2}{2}}dv=\\&=\frac{\sigma}{Q\left(\frac{l-m}{\sigma}\right)}\int_{\frac{l-m}{\sigma}}^{+\infty}v\frac{1}{\sqrt{2\pi}}e^{-\frac{v^2}{2}}dv+\frac{m}{Q\left(\frac{l-m}{\sigma}\right)}\int_{\frac{l-m}{\sigma}}^{+\infty}\frac{1}{\sqrt{2\pi}}e^{-\frac{v^2}{2}}dv=\\&=\frac{\sigma}{Q\left(\frac{l-m}{\sigma}\right)}\int_{\frac{l-m}{\sigma}}^{+\infty}v\frac{1}{\sqrt{2\pi}}e^{-\frac{v^2}{2}}dv+m=\\&=\frac{\sigma}{Q\left(\frac{l-m}{\sigma}\right)}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{l-m}{\sigma}\right)^2}+m\end{aligned} \tag{13}$$

Now, if we pre-assign a certain value γ to the above used tail function $Q(\cdot)$, then the corresponding argument (and so *m*) is determined by the inverse function $Q^{-1}(\gamma)$:

$$Q\left(\frac{l-m}{\sigma}\right)=\gamma\ \Rightarrow\ m=l-\sigma\cdot Q^{-1}(\gamma) \tag{14}$$

so that (13.) can now be rewritten as:

$$m = E(\xi / \xi > l) - \frac{\sigma}{\gamma} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left[Q^{-1}(\gamma)\right]^2} \tag{15}$$

Substituting *m* from (14.) into (15.) results with the following formula for *σ*:

$$\sigma = \frac{l - E(\xi / \xi > l)}{Q^{-1}(\gamma) - \frac{1}{\sqrt{2\pi\gamma}} e^{-\frac{1}{2}\left[Q^{-1}(\gamma)\right]^2}} \tag{16}$$

Finally, substituting the above expression for *σ* into (14.), we obtain the expression for *m*:

$$m = \frac{\gamma \cdot Q^{-1}(\gamma) E(\xi / \xi > l) - l \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left[Q^{-1}(\gamma)\right]^2}}{\gamma \cdot Q^{-1}(\gamma) - \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left[Q^{-1}(\gamma)\right]^2}} \tag{17}$$

So it came out that, after developing formulas (16.) and (17.), we expressed the mean *m* and the variance σ^2 of the Gaussian random variable ξ, by the mean $E(\xi / \xi > l)$ of the truncated cdf, the truncation cut-off and the tabled inverse $Q^{-1}(\gamma)$ of the Gaussian tail function, for the assumed value γ. As these relations hold among the corresponding estimates, too, in order to estimate $\hat{m}$ and $\hat{\sigma}$, we need to first estimate $\hat{E}(\xi / \xi > l)$ and $\hat{\gamma}$ from the sample data:

$$\hat{E}(\xi / \xi > l) = \frac{\sum_{i=1}^{q} \xi_i N_i(\xi_i > l)}{\sum_{i=1}^{r} N_i(\xi_i > l)} \tag{18}$$

$$\hat{\gamma} = \frac{1}{n} \sum_{i=1}^{s} M_i(\xi_i \le l) \tag{19}$$

where N_i and M_i denote the number of occurrences (frequency) of particular samples being larger and smaller-or-equal than *l*, respectively, and $r,s \le n$.
So once we have estimated $\hat{E}(\xi / \xi > l)$ and $\hat{\gamma}$ by (18.) and (19.), we can then calculate the estimates $\hat{\sigma}$ and $\hat{m}$ by means of (16.) and (17.), which completes the estimate of the pdf (9.).

3.3.4 Results of the analysis

Initially, the network traffic was characterized with respect to packet delay variation and packet loss – that were, expectedly, considered as significant influencers on the congestion window. Accordingly, in many tests, for mutually very different network conditions and between various end-points, significant packet delay variation was noticed, Fig. 14.
However, the expected impact of the packet delay variation [7], [13] on packet loss (and so on congestion, i.e. to its window size), has not been noticed as significant, Fig. 15a, 15b.
Still, some sporadic bursts of packet losses were noticed, which can be explained as a consequence of grouping of the packets coming from various connections. Once the buffer of the router, using drop-tail queuing algorithm, gets in overflow state due to heavy

incoming traffic, the most of or the whole burst might be dropped. This introduces correlation between consecutive packet losses, so that they, too (as packets themselves), occur in bursts. Consequently, the packet loss rate alone does not sufficiently characterize the error performance. (Essentially, "packet-burst-error-rate" would be needed, too, especially for applications sensitive to long bursts of losses [7], [9] [10], [13]).

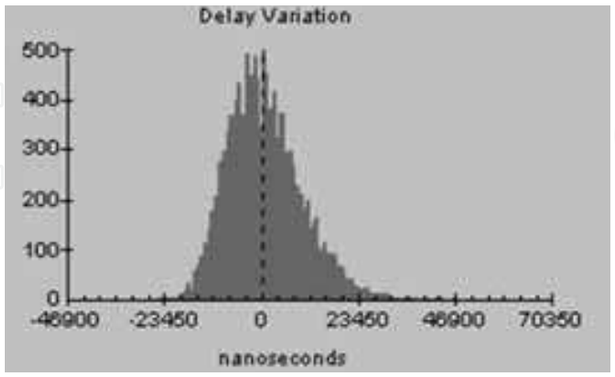


Fig. 14. Typical packet delay variation within a test LAN segment

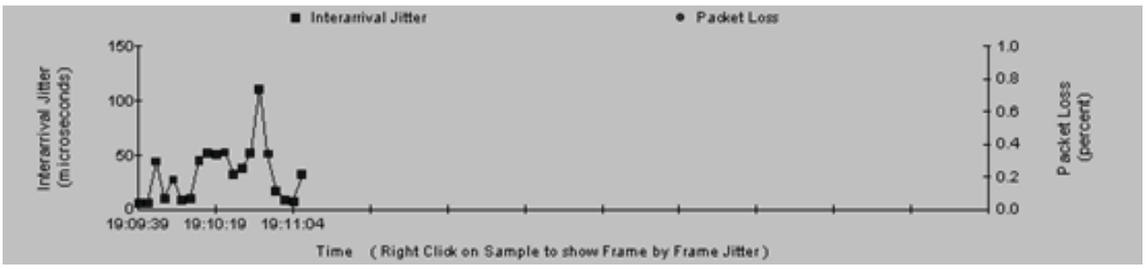


Fig. 15a. Typical time-diagram of correlated packet jitter and loss measurements

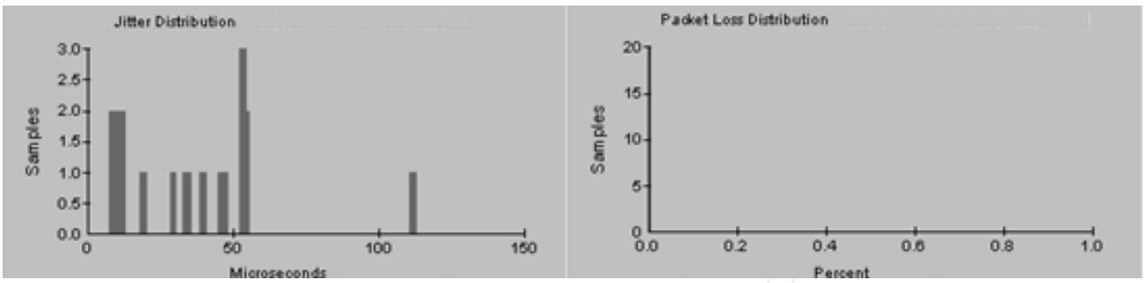


Fig. 15b. Typical histogram of correlated packet jitter and loss measurements

With this respect, one of our observations (coming out from the expert analysis tools we referenced in Section 2) was that, in some instances, congestion window values show strong correlation among various connections. Very likely, this was a consequence of the above mentioned bursty nature of packet losses, as each packet, dropped from a particular connection, likely causes the congestion window of that very connection to be simultaneously reduced [7], [8], [10].

In the conducted real-life analyses of the congestion process stationarity, the congestion window values that were calculated from the TCP PDU stream, captured by protocol analyzers, were considered as a sequence of quasi-stationary series with constant cdf that

changes only at frontiers between the successive intervals [12]. In order to identify these intervals by successive two-sample K-S tests (as explained above), the empirical cdfs within two neighbouring time windows of rising lengths were compared, sliding them along the data samples, to finally combine the two data samples into a single test series, once the distributions matched.

Typical results (where "typical" refers to traffic levels, network utilization and throughput for a particular network configuration) of our statistical analysis for 10000 samples of actual stationary congestion window sizes, sorted in classes with the resolution of 20, are presented in Table 1 and as histogram, on Fig. 16, visually indicating compliance with the (truncated) normal cdf, having the sample mean within the class of 110 to 130. Accordingly, as the TCP-stable intervals were identified, numerous one-sample K-S tests were conducted and obtained the p-values in the range from 0.414 to 0.489, which provided solid indication for accepting (with a=1%) the null-hypothesis that, during stationary intervals, the statistical distribution of congestion window was (truncated) normal.

$Pr(x_i\text{-}20 < x < x_i)$	278	310	624	928	2094	2452	1684	911	478	157	63	21
x_i	30	50	70	90	110	130	150	170	190	210	230	250

Table 1. Typical values of stationary congestion window size

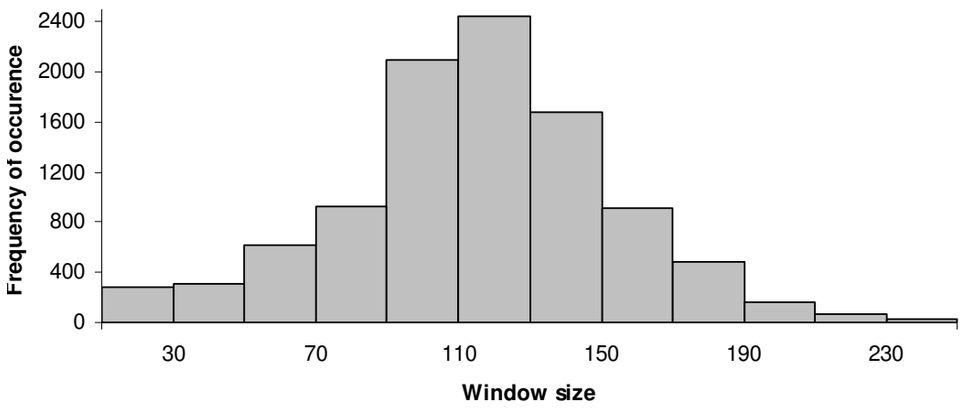


Fig. 16. Typical histogram of the congestion window

As per our model, the next step was to estimate typical values of the congestion window distribution parameters. So, firstly, by means of (19.), $\hat{\gamma}$ was estimated as one minus the sum of frequencies of all samples belonging to the lowest value class (so e.g., in the typical case, presented by Table 1 and Fig. 16, $\hat{\gamma}$=1-278/10000=0.9722 was taken, which determined the value $Q^{-1}(\gamma)$=-1.915 that was accordingly selected from the look-up table). Then the value of l=30 was chosen for the truncation cut-off and, from (18.), the mean $\hat{E}(\xi / \xi > l)$=117.83 of the truncated distribution was calculated, excluding the samples from the lowest class and their belonging frequencies, from this calculation.

Finally, based on (16.) and (17.), the estimates for the distribution mean and variance of the exemplar typical data presented above, were obtained as: $\hat{m}$=114.92 and $\hat{\sigma}$=44.35.

4. Conclusion

It has become widely accepted that network managers' understanding how tool selection changes with the progress through the management process, is critical to being efficient and

effective. Among various state-of-the-art network management tools and solutions that have been briefly presented in this chapter, as ranging from simple media testers, through distributed systems, to protocol analyzers, specifically, expert analysis based troubleshooting was focused as a means to effectively isolate and analyze network and system problems. With this respect, an illustrating example of real-life testing of the TCP congestion window process is presented, where the tests were conducted on a major network with live traffic, by means of hardware and expert-system-based distributed protocol analysis and applying the appropriate additional model that was developed for statistical analysis of captured data.

Specifically, it was shown that the distribution of TCP congestion window size, during stationary intervals of the protocol behaviour that was identified prior to estimation of the cdf, can be considered as close to the normal one, whose parameters were estimated experimentally, following the theoretical model.

In some instances, it was found out that the congestion window values show strong correlation among various connections, as a consequence of intermittent bursty nature of packet losses.

The proposed test model can be extended to include the analysis of TCP performance in various communications networks, thus confirming that network troubleshooting which integrates capabilities of expert analysis and classical statistical protocol analysis tools, is the best choice whenever achievable and affordable.

5. References

[1] Comer, D. E., "Internetworking with TCP/IP, Volume 1; Principles, Protocols, and Architecture (Fifth Edition), Prentice Hall, NJ, 2005

[2] Burns, K., „TCP/IP Analysis and Troubleshooting Toolkit", Wiley Publishing Inc., Indianapolis, Indiana, 2003

[3] Oppenheimer, P. „Top-Down Network Design - Second Edition", Cisco Press, 2004

[4] Agilent Technologies, "Network Analyzer Technical Overview", 5988-4231EN, 2004

[5] Lipovac, V., Batos, V., Nemsic, B., "Testing TCP Traffic Congestion by Distributed Protocol Analysis and Statistical Modelling, Promet - Traffic and Transportation, vol. 21, issue 4, pp. 259-268, 2009

[6]Agilent Technologies, "Network Troubleshooting Center Technical Overview", 5988-8548EN, 2005

[7] A. Kumar,"Comparative Performance Analysis of Versions of TCP", *IEEE/ACM Transactions on Networking,* Aug. 1998

[8] M. Mathis, J. Semke, J. Mahdavi and T. J. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm." Computer Communication Review, vol. 27, no. 3, July 1997

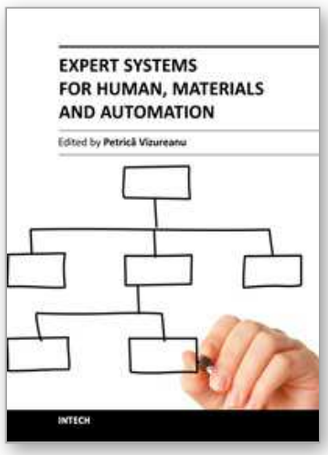
[9] K. Chen, Y. Xue, and K. Nahrstedt, "On setting TCP's Congestion Window Limit in Mobile ad hoc Networks", Proc. IEEE International Conf. on Communications, Anchorage, May 2003

[10] S. Floyd and K. Fall," Promoting the Use of End-to-End Congestion Control in the Internet", IEEE/ACM Trans. on Networking, vol. 7, issue 4, pp. 458 – 472, Aug. 1999

[11] H. Balakrishnan, H. Rahul, and S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts", Proc. ACM SIGCOMM, Sep. 1999

[12] M. Kendall, A. Stewart, "The Advanced Theory of Statistics", *Charles Griffin* London, 1966.

[13] T. Elteto, S. Molnar, "On the distribution of round-trip delays in TCP /IP networks", International Conference on Local Computer Network, 1999



Expert Systems for Human, Materials and Automation

Edited by Prof. PetricÄƒ Vizureanu

ISBN 978-953-307-334-7
Hard cover, 392 pages
Publisher InTech
Published online 10, October, 2011
Published in print edition October, 2011

The ability to create intelligent machines has intrigued humans since ancient times, and today with the advent of the computer and 50 years of research into AI programming techniques, the dream of smart machines is becoming a reality. The concept of human-computer interfaces has been undergoing changes over the years. In carrying out the most important tasks is the lack of formalized application methods, mathematical models and advanced computer support. The evolution of biological systems to adapt to their environment has fascinated and challenged scientists to increase their level of understanding of the functional characteristics of such systems. This book has 19 chapters and explain that the expert systems are products of the artificial intelligence, branch of computer science that seeks to develop intelligent programs for human, materials and automation.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen