



A VitalSoft Whitepaper

Characterizing End-to-End Performance

Introduction

The historical evolution of the Internet and TCP/IP-based computing has emphasized the construction of abstract services (FTP, HTTP, NNTP) whose implementation relies on information hiding. This is a principal reason for TCP/IP's success, because it frees the attention of application programmers from the low-level details of the system. However, this information-hiding philosophy has made it difficult for network professionals to measure and understand the factors determining TCP/IP-based application performance.

With the measurement and characterization of application performance of great interest to application users and service providers, techniques for measuring this performance are of critical importance. This paper examines the measurement techniques employed by VitalAgent and discusses the implications of a "client side" approach to monitoring.

VitalAgent attempts to reveal and clarify the distinct components of an application transaction so that application performance is comprehensible to application users and service providers. This information can then be used to characterize the instantaneous state of each of the components so that the reasons for poor performance can be identified. In addition, this information can provide guidance to users so that they can correct problems to the degree possible.

This paper assumes the reader is already familiar with the Transmission Control Protocol/Internet Protocol (TCP/IP), basic networking concepts and network monitoring. Although experience with VitalAgent is useful, it is not required.

General Principles

The goals of VitalAgent are to:

Measure and characterize networked application performance from the end user's perspective.



Diagnose and isolate problems over the end-to-end path from an application client to an application server.

Isolate these problems to one of the several components along that path.

Perform these functions without requiring the installation of instrumentation in applications, network devices or servers.

Since VitalAgent users will vary significantly in their familiarity with the components of the end-to-end path, forming this characterization requires simplifying raw measurements into high-level, interpreted data. Furthermore it is important to perform such a characterization in a timely fashion.

Throughout this document we will use the example of Web-based applications using the Internet to illustrate the methodology used by VitalAgent. This approach is directly applicable to other application types as well. We have chosen Web-based applications as an example because it is an area of intense interest and development at the present time and because it is a common application type well known to most information technology professionals.

To place the following discussion in context, a typical end-to-end client-server path is depicted in Figure 1. In this figure, a home user ("Enduser") is accessing HTTP-based services from a content provider ("Server") over a Public Switched Telephone Network (PSTN) connection to an ISP. The path from the ISP modem to the server passes through a number of routers and over a number of transmission links of varying capacity.

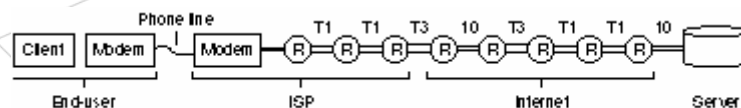


Figure 1. Components of a Client-Server Path

Figure 1 shows the complexity of a typical end-to-end path. As this figure shows, many components are involved; the number and properties of those components are unknown and will vary from connection to connection; and multiple transmission technologies are used. In such a system, most of the components involved have the potential for introducing delays or becoming performance bottlenecks.

Successfully isolating and measuring these individual components is a significant challenge. This task becomes even more challenging while restricting data collection to that available at the client end in the highly variable traffic environment of the modern Internet or Intranet. The



advantages of requiring only client-based measurements are significant enough to merit the effort, however.

This paper outlines a high level view of the measurement methods used by VitalAgent in order to provide a basic understanding of how such challenges can be overcome. We concentrate on the types of data gathered, algorithms used to gather data, and the expressions and formulas used to reduce and transform data. In the process we discuss many practical issues associated with performance measurement in today's Internet/Intranets (such as measurement variability) and methods used in VitalAgent to address them.

Design Principles

The design of VitalAgent's instrumentation at the client adheres to the following important principles:

Minimal intrusion (changes to existing software) in the application, network and server

Minimal perturbation (changes to the system performance as a result of measurement)

Minimal cost of deployment

Minimal resource consumption.

Resource consumption includes memory and CPU consumption and any diagnostic packets issued into the network. While many tools for monitoring client-server paths are resource-intensive, VitalAgent restricts its analysis to the path currently relevant for the application, with the smallest possible use of client and network resources.

Note that VitalAgent is not intended as a replacement of dedicated troubleshooting or performance monitoring and analysis tools such as RMON probes, *tcpdump*, or other tools such as those listed in "Related Works" (Page 1). Further note that VitalAgent's goal is to present useful performance indicators and not precise performance metrics. Therefore, it is more tolerant of errors introduced by noise in the measured system.

One of the major advantages of the client-based approach is the ability to augment less precise network performance measurements with direct local desktop measurements. Examples of these measurements are the time a dial-up connection was established, the duration of such a connection, the negotiated baud rate for the connection, or a web page



retrieval time—values easily obtained at the client via reliable delivery of these sorts of events and their timestamps.

But most performance measurements are uncertain in nature and require additional analysis. The need for analysis is compounded by the constraints of the desktop client environment (for example, lack of high-granularity timers, and the wide variety of operating environments and network environments). It is sometimes impossible to reach the accuracy levels of high-end, dedicated measurement tools. Consequently, VitalAgent provides performance estimates and compensates for these limitations by:

Characterizing many of its measurements in terms of their variability.

Identifying as many data points as possible as candidates for measurement, then filtering out those known to be outside VitalAgent's range of acceptable accuracy, ("outliers").

Deriving indices from raw measurements; these are representative values that indicate the relationship between the current measurement and its recent trend.

In light of these techniques, a key principle employed by VitalAgent is to interpret the current performance measurement against previously observed measurements. Measurements are then used to establish historical baselines in the form of performance distributions. Using these distributions, performance indices are computed and presented to the VitalAgent user in a more useful form; in terms of percentiles rather than absolute quantities.

Session Flow Analysis

VitalAgent calculates and presents performance estimates based on monitoring data flow between the client and the server. Such monitoring is performed at the client end. For the purposes of discussion of these measurements, we define a *session* as a collection of TCP *connections* that the client establishes in order to complete a useful task, such as retrieving a document.

The progress of a TCP connection is well documented [Stev94a]. Most TCP implementations behave in a generally predictable way during connection establishment, normal data transfer mode, and connection teardown [Pax97a]. In general, TCP implementations push on the network to utilize the available network bandwidth capacity in an efficient manner. It reacts to signals such as receiver buffer exhaustion, network congestion and



current network round-trip-time. Good implementations use the available network capacity and deliver data to receivers in a timely way while keeping the network overhead to a minimum.

By observing TCP packet inter-arrival times and other notable events such as retransmissions and gaps in sequence, and by obtaining knowledge of the higher level application behavior, it is possible to learn a great deal about performance conditions of the session's peer and network components. We will term this *session flow analysis*. Based on such analysis, VitalAgent computes the following interesting information about a session:

Transaction time measurements, with breakdown of transaction time into client, network and server components

Congestion magnitude

The bottleneck link speed from the server to the client

End-to-end packet loss

Server throughput estimation

Path Length (number of hops)

Service provider domain identification

Before delving into the details of forming these estimates, however, we first need to discuss the important issue of "measurement vantage point." The **location** along an end-to-end path at which we perform measurement significantly affects both the types of measurements we can make and their accuracy.

Measurement Vantage Point

Characterization of TCP and application performance can be performed at several vantage points in the end-to-end path, including:

As an integral part of the server end

At a point close to the server, perhaps on a host with promiscuous monitoring capability

At a point close to the client, once again with promiscuous monitoring capability

At any arbitrary point between the client and the server

At the client desktop



Measurement accuracy, the amount of resources required for analysis, and the availability for instrumentation all vary for each type of vantage point. Measurements that are tightly integrated with the server end software/hardware require instrumenting mission-critical servers. Such instrumentation always runs the risk of degrading server reliability and performance. While basing measurements on analysis at a point close to the server (through promiscuous monitoring) overcomes this, it is often difficult to access a point where such instrumentation is available. Furthermore, for operations involving dominant flow from the server to the client, it is more relevant to observe TCP characteristics at the client end.

End-to-end analysis based on measurements at an arbitrary point along the client-server path suffers from difficulties in gathering all the data promiscuously, identification of sessions, asymmetric paths, and combining network datagrams into transport and application level data streams. In addition, such vantage points in the Internet/Intranet are often inaccessible or provide far too much data to be analyzed.

The TCP flow-based measurements of VitalAgent are obtained from observations at the client end of the session. Analysis at the client end has several benefits:

The dominant flow is frequently from the server to the client allowing VitalAgent to characterize the more dominant direction.

Access to application-level flow is immediately available. (As an example, one can correlate client-side interaction by pairing network layer segments with the corresponding socket-layer send and receive stream.)

The results of data analysis can be made available to the end user in a very timely way. Results of analysis do not have to be shipped across the network for the end-user to gain visibility to the current performance conditions.

There is usually a fair amount of computing resources available for analysis during the time the client is collecting TCP and application-level data buffers.

Analysis of TCP and application flow is restricted to the single end-to-end path that is being observed.

If it is possible to observe and correlate measurements from multiple vantage points, accuracy can be improved; however, there are technical challenges in correlating raw data from multiple sources in real-time. As



an example, the *tcpanaly* [Pax97b] utility analyzes traces recorded by *tcpdump* at both the server and the client side. Such analysis, however, is difficult to perform in real-time because the two sets of measurements must be compared in detail. If measurements are to be made only from a single vantage, the client side of a session is for many purposes the best location for observing a TCP flow.

Latency Measurements

The first type of TCP-flow based measurements made by VitalAgent is latency measurements. The progress of a TCP three-way handshake provides an opportunity for measuring network latency. In general, packet exchange sequences are used in the establishment of Round-Trip-Time (RTT) estimates and retransmit timers within a TCP implementation. For client-side measurement, this estimation is more difficult once a connection is established, because it is necessary to match the time at which a client sends an ACK with the time at which the server's data segments are sent in response to having received the ACK. There is almost never an opportunity to access the timing of this "data triggering" condition at the server, if we can only observe packets at the client.

Most TCP implementations respond with an ACK for the SYN rapidly during the establishment of the connection. Such processing is often done within the kernel without an application-level context switch. By measuring latency of session establishment, VitalAgent can estimate network latency in the client server path without unduly clouding this estimate with application-level delays at the server.

Figure 2 is a simplified diagram of the progress of a normal TCP connection. The vertical lines represent time, advancing from the top to the bottom. The client (for our purposes a browser using the HTTP protocol) initiates a connection to the server at time $TC1$ using a TCP SYN packet, which may transit through a large number of hops before its reception by the server at time $TS1$. The acknowledgment for the SYN is sent by the server at time $TS2$. Given the assumption that the server processes the request in a timely manner, the difference in times ($TS2 - TS1$) is assumed to be very small. Because $TS2 - TS1$ is small, the difference in times observed at the client $\Delta T1$ is a good estimator of instantaneous network latency.

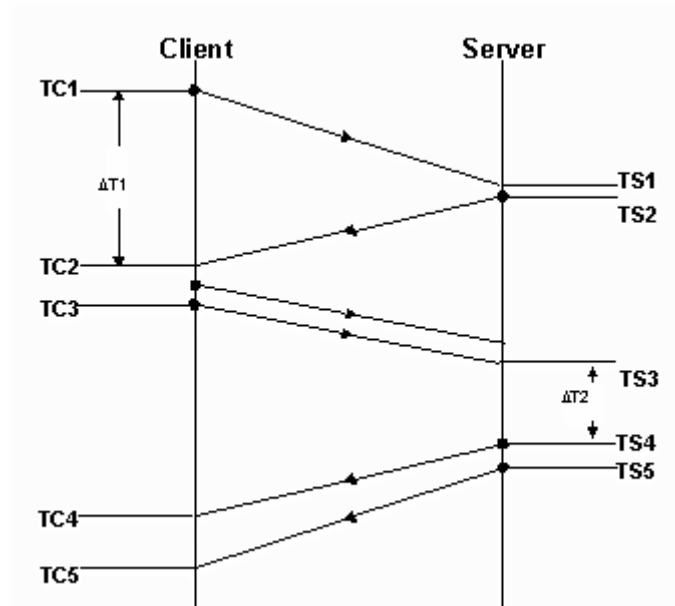


Figure 2. Packet Exchange for HTTP Session

Client Delay after Session Establishment

After connection establishment, an HTTP client prepares to send an HTTP Get Method for a uniform resource locator (URL). The request usually fits within a single IP packet.

The delay ($TC3-TC2$) for the client to construct this request and send it out on its network interface is small and easily measured at the client. In practice, upon receipt of a SYN-ACK from the server, the client sends a TCP packet of length zero with an acknowledgment. This is followed by the data packet containing the HTTP Get Method. VitalAgent takes care to note the timing associated with the actual data packet, but not the earlier ACK, and use it in its timing measurements. This method of measurement is extensible to other TCP/IP-based applications as well.

Server Delay Estimation

An important TCP data flow based measurement provided by VitalAgent is server delay for processing a request. At the server end, the HTTP **Get** Method is received at $TS3$ and a response is sent after *delta T2* time, at $TS4$.

This response is subject to a number of factors:



The nature of the HTTP Get Method. If the request is for dynamically created content or an expensive database operation, the delay is considerably longer.

The current processing load on the server. If the load on the server is high, the time to serve this client's request is increased.

The speed of the processor. If it is a slow processor, the delay can be high even with a lower load.

The network load at the server. The load on the segment that supports the server can impede performance.

When a response reaches the client, the delay measured between $TC4$ and $TC3$ is an indicator of both the network delay and the delay introduced by the server. Given that the initial delay ($TC2-TC1$) was an estimator of network delay alone, the difference in the two delay measurements can be used to characterize the server. The index also needs to account for differences in packet sizes as well as queuing delays on the forward and reverse paths.

Delay Measurements per Http Session

Most client requests to a server involve the retrieval of multiple components of an HTML page. Such a request is frequently referred to as an *HTTP session*. Some browsers use multiple concurrent connections to accelerate the retrieval up to certain point [Pad95]. While this appears attractive, the TCP protocol-specific parameters (for example, the Round-Trip-Time estimation and congestion window) are not shared among such connections and is in some sense an ineffective use of the protocol¹. However, the use of multiple connections also affords an opportunity to observe several connection establishment sequences. These delays can be analyzed together to form a better estimate of current latency in the network.

This section analyzes the impact of multiple connections per HTTP session and the distribution of latency measurements per document retrieval. In order to illustrate the nature of multiple connections, we present the SYN-ACK sequences of connection requests for the home page retrieval from a large operational web site, which we shall call *exampleserver.com*. While here we focus on a single example session, large scale studies have been performed by others [Mog95].

Table 1 shows the time at which a SYN was sent and the corresponding SYN-ACK was received. Notice that it takes 12 independent connections to



download the contents of this page. Also note that because the browser configuration allowed only four concurrent connections (the default), there are never more than four connections overlapping at any point. These measurements were for a bottleneck link speed of 256 Kbps (with a system connected to a local area network (LAN), in turn connected to the Internet via a fractional-T1 link).

Connection	SYN	SYN-ACK	RTT
1	0.000	0.088	0.088
2	0.819	0.979	0.160
3	0.830	0.981	0.151
4	0.859	0.983	0.124
5	1.213	1.338	0.125
6	1.300	1.389	0.089
7	1.364	1.469	0.105
8	1.557	1.646	0.089
9	1.581	1.682	0.101
10	1.604	1.699	0.095
11	1.635	1.727	0.092
12	1.835	1.925	0.090

Table 1. Connection Establishment Times

As Figure 3 indicates, if the RTT of the SYN and ACK were plotted against the start time of these connections, these times fall within a fairly close range. Obtaining the median of these RTTs is useful for the following purposes:

As a robust characterization of network round trip delay

As a good baseline for the connection establishment phase

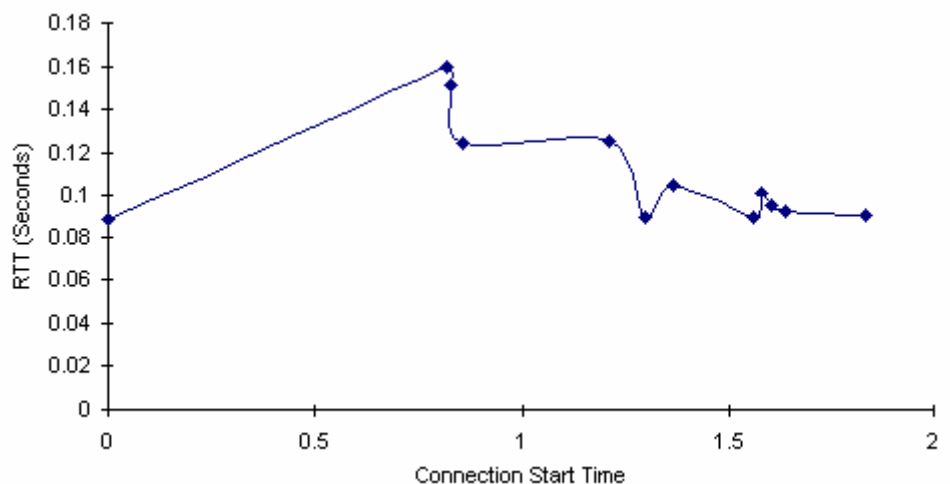


Figure 3. Graph of Connections and their Round Trip Time (RTT)



Note that while the use of the minimum RTT is attractive, limited clock granularity sometimes renders a value that is significantly less than the actual RTT. VitalAgent uses a baseline technique to overcome these exceptions.

It is also informative to build a histogram of the distribution of RTTs, as shown in Figure 4. The connection establishment times are shown in the form of a histogram for the retrieval of the <http://www.exampleserver.com> page at various points in the day. The total number of HTTP page retrievals was 50, with the local caching of images and other components disabled. Using this historical distribution, predictions about network congestion levels can be made because the variations in latency measured above in part is a function of network congestion.

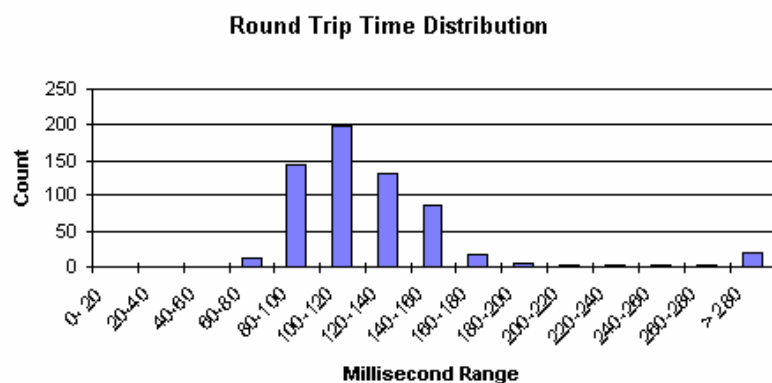


Figure 4. Histogram of Round Trip Time (RTT) Distribution

The following observations can be made:

The distribution of RTT displays a small percentage of connections that were greater than 300 milliseconds, which tended to push the mean to higher numbers. As a result, the median of the graph is a more robust indicator of typical RTT, as opposed to the mean.

While it is computationally resource-intensive to derive and maintain a median (especially if each server access by the client were to be tracked), VitalAgent uses a scheme to compute a value close to the median, as described in Section "[Median and Percentile Measures](#)".

Given an observed RTT (as indicated by the initial SYN-ACK sequence), its percentile location in the histogram of existing RTTs can be used to characterize the various network effects (such as congestion and queue depths of intermediate routers) that contributed to that RTT.



For each connection observed above, the server delay ($TS4-TS3$) was also noted. As mentioned earlier, this measurement is indicative of server load. However, VitalAgent only has access to the measure $TC4-TC3$ because measurements are made only at the client end. An indication of $TS4-TS3$ is possible for the connections noted above, if the network RTT is separated out from the $TC4-TC3$ measure, by subtracting the estimated RTT from $TC4-TC3$.

Table 2 illustrates the computation of this delay. In this example, a median network RTT of 92 milliseconds was used as the network round trip delay.

TC3	TC4	TC4-TC3	TC4-TC3 - RTT
0.116	0.347	0.231	0.139
1.168	1.292	0.124	0.032
1.181	1.306	0.125	0.033
1.188	1.360	0.172	0.080
1.400	1.549	0.149	0.057
1.429	1.572	0.143	0.051
1.504	1.616	0.112	0.020
1.681	1.789	0.108	0.016
1.713	1.840	0.127	0.035
1.729	1.847	0.118	0.026
2.013	2.166	0.153	0.061

Table 2. Delay Measurements at the Client End of TCP Sessions

The distribution of the $TC4-TC3-RTT$ measurement can be used to characterize the load on the server because it potentially reflects how quickly the server is able to respond to the HTTP Get method.

Figure 5 indicates the delay ($TC4-TC3-RTT$) for eleven connections that were completed for retrieving the <http://www.exampleserver.com> page. For a correlation of this type of measurement and busyness of servers, see [Mog95].

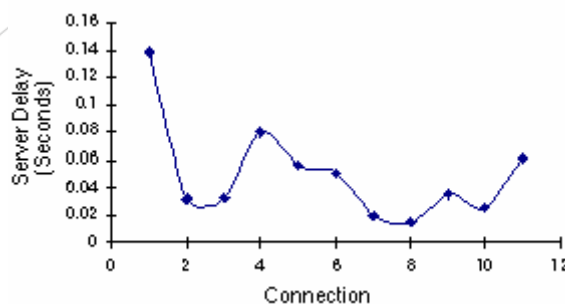


Figure 5. Graph of Server Delay Measurement

A histogram of this server delay reveals a distribution weighed more heavily to larger delays. Therefore, VitalAgent uses the median and



percentile as robust measures to characterize server delay, rather than the mean.

Median and Percentile Measures

The RTT measurements and server latency measurements have been shown not to follow conventional statistical distribution (such as Poisson or Gaussian) [Pax97]. An unfortunate consequence of this is that mean latency is rarely a good indicator of typical behavior of a component. A more appropriate measure of typical behavior is the median. If the complete distribution is available, the percentile of a sample can be used to provide the probability that an observation of this size or less would be seen in practice.

Given a sample point, computing its exact percentile in a distribution of previously seen samples is a computationally intensive endeavor. This is because one has to maintain all data points in a sorted fashion and allow for insertion of new data into this list and then determine its location in the list. To expedite this process without these computational requirements, VitalAgent uses the following technique.

Data points are maintained in a histogram of a certain bucket size. For each new data point, VitalAgent identifies its bucket and updates the number of occurrences of data points in that bucket range. The percentile of that data point is the sum of all counts in buckets prior to the bucket range of that data point, plus a percentile number assuming a near linear distribution of data points in its bucket.

Congestion Measurements

Congestion may be the most complex TCP based flow measurement that VitalAgent estimates. In most cases, the capacity of links between the client and the server varies from link to link. It is common to see a client connected by a slow-speed modem link to a modem bank which feeds into a high speed LAN. The data then flows through a router that has one or more WAN links (such as a 1.5MB/sec T1 link). The differences in link capacity at each hop in the client-server path can cause congestion, as can competing traffic.

For most dial-up connections, the client is typically on a link with less capacity than all other links on the path to the server. Consequently, a smaller link feeds into a larger link when data is transferred from the client to a server. Therefore, the likelihood of congestion occurring downstream from the client due to this factor is low. On the other hand,



when responses are sent by the server to client there is a high probability that the data will be queued at a device waiting to be pushed through the slower link.

The TCP slow-start algorithm [Jac88] adapts to these variations. The Congestion Window (*cwnd*), required for all TCP implementations, limits the amount of data a connection can have in flight, and evolves over the course of the connection. When it increases to a point at which packet loss occurs, the sending TCP infers that the loss occurred due to congestion, and diminishes *cwnd* by half to effectively cut its sending rate in half.

While the extent to which *cwnd* can grow is also limited by the receiver's window advertisement, our observations of HTTP transactions indicate that the receiver's TCP window is almost always fully available and is rarely a limiting factor. Thus, *cwnd* is almost always limited by current load conditions of the network.

At the receiving end, the impact of the congestion window is visible in how quickly TCP segments appear. In particular, at the receiving end it is often possible to identify TCP segments that the sender most likely sent back-to-back. At the receiving end, however, if there is a long delay between such packets, it is an indicator of competing traffic entering between the packets during their transit across some hop between the sender and the receiver.

In Figure 2, the server sends two segments at TS4 and TS5 after determining that *cwnd* is at least two. These two packets are then subject to network delay and arrive at TC4 and TC5 at the client. Given that these two segments were sent very close in time to each other, their arrival times at the client are an indicator of congestion factors in the network.

If there was competing traffic that managed to intervene between these two packets, its presence in the queues of the routers will increase the TC5-TC4 delay differential (it is even possible for TC5-TC4 < TS5-TS4, although this is rare [Pax97a]). While preservation of this delay and its propagation across all subsequent segments is not guaranteed, continued and persistent occurrence of this delay is an indicator of competing traffic causing congestion.

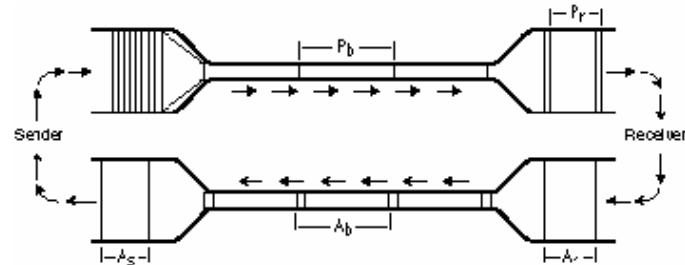
Bottleneck Speed Measurement

The maximum available bandwidth end to end, or "bottleneck link speed" is a critical index for client/server performance. It is interesting to use the



following two approaches to study the impact of differences in the capacity of various links from server to client.

Packet Stretch



The first technique is to measure *packet-stretch* which measures the extent to which the inter-packet gap between two consecutive packets gets stretched by the bottleneck. If two or more packets happen to be queued up in a router upstream to the bottleneck link, they travel the bottleneck link in back-to-back sequence, which introduces an inter-packet gap inversely proportional to the speed of the bottleneck link. Figure 6 illustrates the packet flow through a bottleneck link (adapted from [Jac88]), where P_b is the extent to which data occupies the bottleneck link and A_b is the inter-packet gap introduced by the upstream router. The bandwidth measurement tool BPROBE [Cro96, Car96] uses this technique for measuring current and base bandwidth levels.

Figure 6. End-To-End Packet Flow through a Bottleneck Link

A variation on the above technique is to identify TCP segments that the server sends that are known to be back-to-back. An observation of inter-packet gaps (P_r in Figure 6.) at the client for these packets will then provide identification of the bottleneck link speed. A major advantage of this variation is that VitalAgent can passively use the data packets from the server itself, without adding additional measurement traffic.

[Cro96] describes the following practical problems in measuring the base bandwidth using this technique—queuing failure, competing traffic, loss of packets, and downstream congestion—along with solutions for overcoming them.

[Cro96] also discusses robust filtering techniques to estimate bottleneck link speed, as does [Pax97a], which also details a number of additional estimation pitfalls.



Packet Size Variation Technique

The second technique is to send ICMP packets of varying sizes to the target system. Bottleneck link speed is then estimated based on the variations in propagation delay. The *pathchar* utility obtains a hop-by-hop bottleneck link speed using this technique [Jac97].

Packet Loss and its Impact on TCP Throughput

TCP implementations on both the sending and receiving sides cooperate to detect packet loss and quickly recover from it. If the sending side does not receive an acknowledgment for data that was sent within its timeout period, it progressively retransmits those unacknowledged segments. In cases where the receiving side detects an apparently lost segment, it has the ability to detect a packet loss well in advance of the sender's retransmit timer and is required to immediately resend its previous acknowledgment. The sender, upon detection of a sufficient number of these duplicate acknowledgments, retransmits the lost segment.

Figure 7 illustrates a typical flow pattern when there are packet losses and the TCP implementation on the sender side implements slow-start, congestion avoidance and fast recovery algorithm, as described in [Jac88, Stev94a]. This sequence plot shows observations obtained from passive monitoring on a third system, placed close to a client.

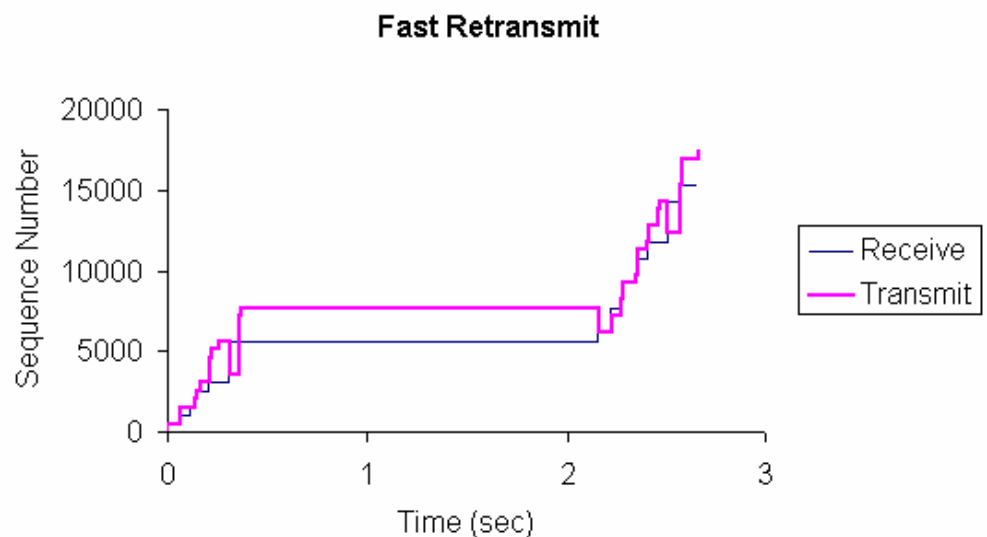


Figure 7. Packet Loss Impact

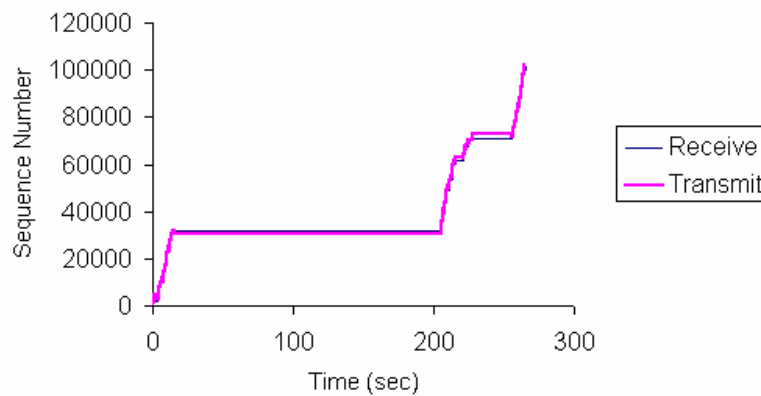
As Figure 7 depicts, there was a packet loss at 0.206 seconds. The receiver began to send acknowledgements only up to sequence 3073,



while the sender continued to send segments up to 0.315 seconds (sequence 5633). The sender then detects that the segment beginning at 3074 was lost due to receipt of three duplicate acknowledgements for 3073, and retransmits. Since only one segment was lost, the receiver acknowledges up to 5633 and transmission of new data continues. Similarly, after a loss of acknowledgement at 2.15 seconds we see that data transfer occurs via fast retransmission; i.e., the sending side TCP retransmits what appears to be a missing segment without waiting for the retransmission timer to expire.

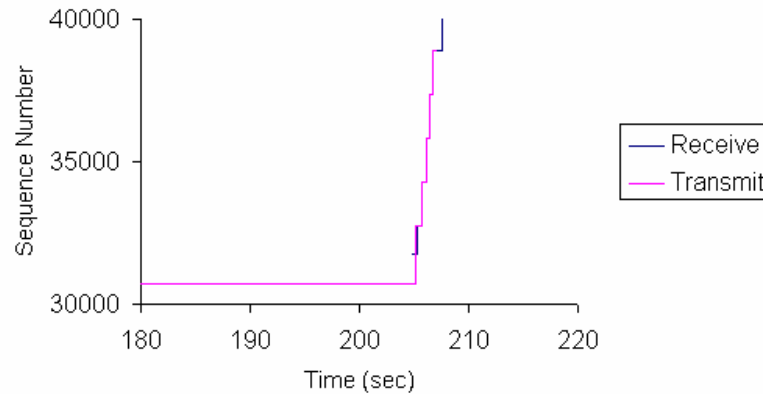
The following two figures show the behavior after a timeout period. When the sending side does not receive acknowledgements for a long duration (which is self adjusting based on TCP's round trip time estimation) it begins to retransmit those segments that have not been acknowledged. In the first figure, TCP recovers from a timeout at about 200 seconds. The behavior at around that time is expanded in the second figure.

TCP Timeout Sequence





Recovery after timeout



As demonstrated in the two examples above, by observing the progression of received sequence numbers at the client and the ACKs that were sent, one can determine that there was a loss or timeout from the server to the client. Also, if the ACKs were dropped, one can determine that there was a loss from the client to the server. Such a scenario will present a different progression of sequence numbers and its ACKs. Therefore, an observation of duplicate received sequence and its duplicate ACK at the client end reveals that an ACK was lost.

For a complete discussion of packet loss and performance degradation, a discussion of exponential backoff in the face of repeated loss, and how cwnd is effectively halved upon loss and only slowly grows back, is important. These items have a major impact on performance, much more so than the single retransmission itself, yet are outside the scope of this document. To avoid end-user confusion about the meaning of packet loss, VitalAgent does not expose these raw measurements in its user interface.

Server Throughput Estimation

The server throughput of a client-server session is defined as the number of bytes of application-level data that is transferred from the server to the client during the connection. It is an average of the rate of transfer over the time period of the session. In the context of an HTTP session, it includes the bytes sent and received for all TCP connections from the time of connection establishment to the time connection tear-down occurs. In this context, the throughput measurement considers only the application-level data, which is delivered, and does not use any transport or network level data for the throughput calculation. For example, if there were a



large number of retransmitted TCP segments, its data is not included in the throughput estimation.

The measurement of throughput of the session is merely the advance of the receiver's acknowledgment of the sender's data. The time at which these acknowledgments are sent provides the time base for the throughput computation. If a session has reached steady state (for example, slow-start has completed and the congestion window on the sender has reached the point of packet loss) and the network is not lossy or congested, then the throughput delivered will be fairly steady.

Figure 8 shows the rate at which network and application data was delivered for access to <http://www.exampleserver.com>. (Note that the network data line is the line with higher data rates).

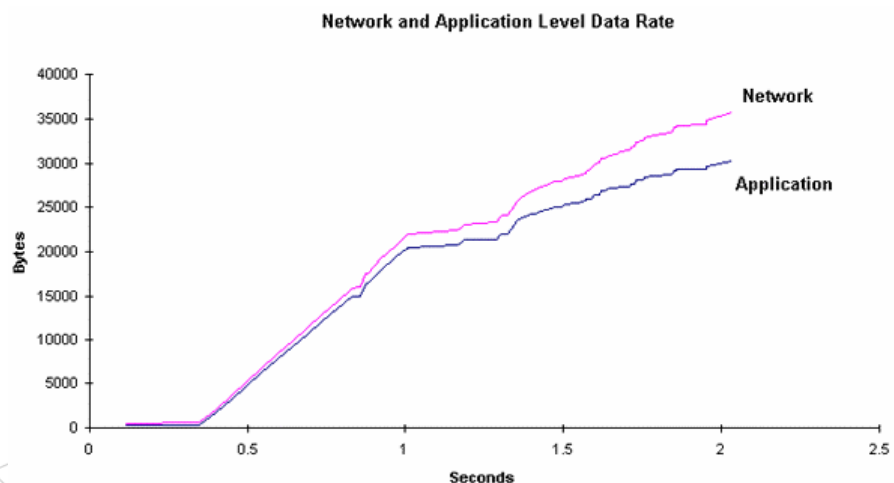


Figure 8. Network and Application-level Data Rate

The ratio of application-level data rate and the network-level data rate is an indication of how well the protocol is operating given the current congestion levels and available network bandwidth. The network-level throughput includes the following: (1) the network-layer bytes (TCP and IP headers), (2) the bytes present in retransmitted TCP segments, and (3) the zero-data ACK packets sent in the direction of interest (which contribute TCP and IP header bytes). In contrast, the application-level throughput is a measure of user data bytes that the application received.

Consequently, the network-layer throughput is affected by several factors: the TCP overhead and the inefficiency of the TCP protocol, as well as the impact of packet loss and congestion on the TCP session.



The overhead bytes are easily measured by counting the number of TCP segments and adding a constant for packet headers (While the overhead can vary based on factors such as IP options, for the most part it is constant.). The impact of packet loss and congestion on the session is then easily estimated by using the difference, as follows:

$$C = Tn - Tg - \frac{(N * S)}{t}$$

where C is the throughput lost due to retransmissions and network overhead, Tn is the network-level throughput, Tg is the application throughput measured, N is the number of TCP segments, and s is a constant number of bytes in a TCP header (usually 40 bytes).

An effective throughput measure can be computed by comparing the application-level throughput achieved to the network-level throughput currently achievable on the path. In addition to the protocol efficiency and the congestion factors, this measure includes the impact of delays introduced at the server and client ends. Given these two throughput measures, effective throughput is computed as follows:

$$EffThroughput = \frac{ApplicationThroughput}{AvailableNetworkThroughput} * 100$$

Number Of Router Hops

In addition to application and TCP flow-based measurements, VitalAgent estimates the number of router hops along the path from the server to the client. This measurement is useful in gaining an understanding of the location of the server with respect to the client. While there is not usually a strong correlation between the number of hops and the delay or throughput measurements of the on-line session, it is useful in locating and positioning the demarcation points of a session.

Traditionally, the number of router hops is measured through a technique used by the *traceroute* utility, which is based on increasing the IP Time To Live (TTL) field, as explained in [Stev94b]. Dedicated Web servers also provide well-advertised gateways to the traceroute command to provide hop count measurements from different Internet locations[Boa96]. This dedicated server can then perform a remote trace route from their location and report the results back to the client.

While the *traceroute* technique is useful for obtaining the number of hops, it has two major drawbacks:



The amount of time it takes to obtain its results

The expense in the number of packets transmitted over the network.

To overcome these limitations, VitalAgent typically obtains its number of hops from an observation of TTL fields in the IP packets received at the client. Such analysis includes variations in the nature of TTL assignments at the sending entity. VitalAgent understands a wide variety of server TCP/IP implementations in the area of TTL assignment and adjusts its estimate accordingly. Note that since TTL is observed at the client end, the number of hops will reflect the hops on the return path.

Identification of the Service Provider Domain

In addition to the number of router hops in the path, VitalAgent also identifies the different service provider portions of the client-server path. These portions of the network path are referred to as the *service provider domains*. Frequently, a client-server path through the Internet is likely to traverse several different service providers. Consequently, it is important to identify the various service providers and the hops associated with each provider.

VitalAgent's demarcation algorithm splits the routers in the client-server path into the following three groups:

Intranet routers.

Internet Service Provider (ISP) routers.

The rest of the Internet.

The routers that belong to the Intranet and ISP are identified through two demarcation points: (1) the ISP entry demarcation point (the ISP entry router), and (2) the ISP egress demarcation point (the ISP egress router).

Figure 9 shows a possible arrangement of the service provider domains.

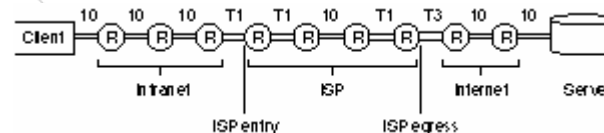


Figure 9. Identification of the Service Provider Domain

VitalAgent identifies ISP entry and egress demarcation points as follows:

By using a local database of router prefixes, and by using the results from an initial discovery or exploration of paths to various points in the Internet.



It limits such exploration to the entry and egress points of the ISP of the end user. For entries that may not be present in such a database, VitalAgent performs a reverse name lookup to obtain a fully qualified domain name and analyzes its results (e.g., core1.Atlanta.mci.net belongs to MCI).

Application-Level Problem Determination

In addition to session-based measurements, VitalAgent measures and performs problem analysis based on its observation of application-layer behavior. Details of application layer monitoring is beyond the scope of this document.

Other Measurements

In addition to application and TCP flow based measurements, VitalAgent computes and reports the following measurements among others:

The number of HTTP page requests

The number of times a page request failed

The number of times dial-out activity was initiated

The call completion rate

The nature of modem dial-out error conditions

Application availability

Connection errors

Transaction failures

However, because these measurements are direct observations of client activity and do not require significant analysis, they are not discussed here.

Accuracy Issues

This section discusses the following factors that affect the accuracy of VitalAgent's performance estimates:

Spot measurement and averaging for a page

Clock Granularity

Server load impact

Client load impact



Impact of proxy servers

Asymmetric routes

Many of these issues are overcome by historical trending and baselining of these performance estimations; we give some examples of how VitalAgent does so below.

Spot Measurement and Averaging for a Page

In general, estimates based on a single diagnostic packet are not reliable due to fairly large fluctuations in congestion levels of routers and other intermediate devices along the path between a client and its server. At each intermediate point, packet arrivals are generally distributed in many complex patterns, as described in [Cro96, Lei94, Pax95]. The measurements of RTT to the destination server overcome these uncertainties by averaging the RTT over multiple connections for the same application. Because VitalAgent is concerned with presenting performance indicators rather than precise metrics, the noise introduced by these effects is tolerable.

Clock Granularity

Many VitalAgent estimates are based on the ability to observe the time delay between events of interest. In some cases, the time delay between these points is very small. This is especially true under the following conditions: (1) there are no long hops between the client and the point of interest, and (2) the bottleneck link speed between the client and the server is very high, making some of the delay measurements fall in the microsecond range.

Most client operating systems either do not provide a clock of microsecond granularity or cannot guarantee the timeliness of a context switch timestamping to the desired level of accuracy. Fortunately, under these conditions, the performance estimation can tolerate significant uncertainty because the client-server interaction also completes fast and produces a high level of online experience. Once again, because the goal of VitalAgent is to present performance indicators, we consider it acceptable if, when the online experience is good, it fails to timestamp precisely.

Server Load Impact

Excessive load conditions on the server can have an impact on certain VitalAgent estimates. In particular, network-level throughput



measurement assumes that the server is capable of responding to a SYN with an immediate SYN-ACK. If this is not the case, there is a potential to make an incorrect estimation of network throughput. This is mitigated by using values from multiple TCP connections that were established as part of an application session for retrieving the contents of a single page. However, if server load persists across multiple sessions, this approach may provide only a rough estimate.

Client Load Impact

There is a direct correlation between the current load conditions at the client and the performance estimates that are computed. In general, most clients are limited by the network capacity of their links. This is especially true where there is a slow-speed, dial-up connection from the client. However, observation of various estimates when the client is busy with local operations (such as formatting a local disk while the HTTP client is issuing requests) biases the estimates to quite an extent. One way to mitigate this effect is to ignore data points that are derived while the CPU load is high.

The last three problem areas are overcome using a local database containing service provider-specific router addresses.

Impact of Proxy Servers and N-Tier Architectures

When proxy servers are present, network measurements reflect only the path between the client and proxy server. The network delay and the destination server delay are combined into the delay due to the (proxy) server. In most situations, the proxy server is either within an Intranet or is at the boundary of an Intranet, as shown in Figure 10.

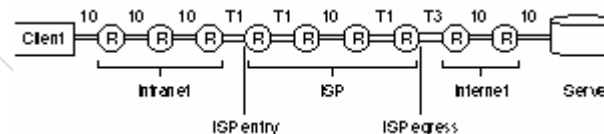


Figure 10. Presence of Proxy Servers in a Client-server Path

The delays measured through passive observation of application-layer data arrivals will include the delay in fetching data from the destination server, if content was not fetched from the proxy server. Thus, the main effect of proxy servers is to attribute wide-area network delay to the server delay.



An analog to this is the case of an n-tier server architecture, where a host mainframe, for instance, is front-ended by a UNIX server. In this case the UNIX server has the same effect as the proxy server. End to end calculations will still be correct, however network time between the UNIX Server and mainframe host will be classified as server delay. Network measurements will reflect only the path between the client and the first-tier server.

Asymmetric Routes

Current Internet routing practices often lead to asymmetric routing of packets in the Internet. This condition also occurs in corporate networks and Intranets. By asymmetric routing, it is meant that the forward and return paths of an end-to-end client server connection traverse different sequences of router hops. A study by [Pax96] indicates that such asymmetry is on the rise. For most measurements VitalAgent reports its estimates based on the route that exists on the return path (e.g. from the HTTP server to the client). This manifests in the following way:

The number of hops shown by VitalAgent is the number of routers in the path from server to client.

Bottleneck link speed and congestion estimates determined by VitalAgent are for the return path.

From the perspective of the client, asymmetric routes do not negatively influence estimates of performance. This is because these estimates reflect the path from server to client and this is the dominant direction of HTTP data flow and the dominant path for most client-server application flows as well.

Related Works

The Cooperative Association for Internet Data Analysis (CAIDA) has compiled a substantial collection of measurement tools [CAI97]. The following tools are described: *Ping*, *traceroute*, *Tcpdpriv*, *NetSCARF*, and *Treno*.

In addition, the following information is also publicly available.

Exchange Point Measurements. Measurement of actual provider-to-provider exchange point statistics are computed for several large providers. For each physical network media type, topology and device, the San Diego Supercomputer Center obtains these statistics at the New York NAP and plots the data exchange information (packets in, packets out,



bytes in, bytes out, packet in discards, and packet out discards) into HTML pages for viewing. Such measurements reflect the actual usage of the exchange point [MFS97] and can provide an accurate measure of congestion levels on the Internet backbone. It also identifies points where there were failures of an exchange path.

IPMA Tools. The Internet Performance Measurement Analysis (IPMA) project has developed several tools for measuring Internet performance for use by ISPs. (1) *AS Explorer* is a utility that provides routing exchanges between the large providers (at the Routing Arbiter's location). (2) The NetNow daemon provides latency and loss statistics by performing a User Datagram Protocol (UDP) spray to various remote probes/daemons located in the Internet. It controls the data spray sending times to reflect a Poisson distribution. (3) Routing instabilities are constantly monitored and reported at the major ISP exchange points by a utility called the Route Flap Statistics Generator.

Footnotes

1. Note that with widespread deployment of HTTP 1.1 and pipelined requests, the approach of using multiple connections may become obsolete. Performance implications of this are studied in [Nie97]. [Return to citation.](#)

Citations And References

[Boa96] Index of TraceRoute Web Servers, <http://www.boardwatch.com/isp/trace.htm> and <http://www.ra.net/tools/trace.html>.

[CAI97] CAIDA Measurement Tool Taxonomy, <http://www.nlanr.net/Caidants/meastools.html>.

[Car96] Robert L. Carter and Mark E. Crovella, *Measuring Bottleneck Link Speeds In Packet-Switched Networks*, TR-96-006, Boston University Computer Science Department, March 15, 1996. Slightly modified version appeared in *Performance Evaluation*, Vol 27&28, 1996.

[Cro96] Kihong Park, Gi Tae Kim, and Mark E. Crovella, *The Effects of Traffic Self-Similarity on TCP Performance*, Boston University Computer Science Department, April 1996.

[IPM96] IPMA Tools, Internet Performance Measurement and Analysis Project.

[Jac88] V. Jacobson, "Congestion Avoidance and Control," Proc. of SIGCOMM '88, 1988.



[Jac97] PATHCHAR, A tool to infer Characteristics of Internet Paths, Van Jacobson, Lawrence Berkeley National Laboratory, April, 1997, <ftp://ftp.ee.lbl.gov/pathchar/msri-talk.ps.gz>.

[Lei94] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, *On the Self-Similar Nature of Ethernet Traffic (Extended Version)*, IEEE/ACM Transactions on Networking, 2(1), pp. 1-15, Feb. 1994.

[MFS97] *Exchange Point Information*, MAE East - Gigaswitch #1 - 5 day composite, MAE West - MFS FDDI - 5 day composite.

[Mog92] J. C. Mogul, *Observing TCP Dynamics in Real Network*, Proceedings of ACM SIGCOMM '92.

[Mog95] J. C. Mogul, *Network Behavior of a Busy Web Server and its Clients*, DEC WRL Research Report 95/5, October 1995.

[Nie97] Henrik Frystyk Nielsen (W3C, MIT), Jim Gettys (W3C, Digital), Anselm Baird-Smith (W3C, INRIA), Eric Prud'hommeaux (W3C, MIT), HÅkon Lie (W3C, INRIA), Chris Lilley (W3C, INRIA), *Network Performance Effects of HTTP 1.1, CSS1 and PNG*, Proceedings of SIGCOMM '97.

[Pad95] Padmanabhan, V.N., *Improving World Wide Web Latency*, 1995, UCB/CSD-95-875, Computer Science Division, University of California, Berkeley, <http://www.cs.berkeley.edu/~padmanab/papers/masters-tr.ps>.

[Pax95] V. Paxson and S. Floyd, *Wide-Area Traffic: The Failure of Poisson Modeling*, IEEE/ACM Transactions on Networking, 3(3), pp. 226-244, June 1995.

[Pax96a] V. Paxson, *Towards a Framework for Defining Internet Performance Metrics*, Network Research Group, Lawrence Berkeley National Laboratory, February, 1996. Proceedings of INET '96.

[Pax96b] V. Paxson, *End-to-End Routing Behavior in the Internet*, Proceedings of SIGCOMM '96, pp. 25-38, Aug. 1996.

[Pax97a] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*, Network Research Group, Lawrence Berkeley National Laboratory, May, 1997, <ftp://ftp.ee.lbl.gov/papers/vp-thesis/dis.ps.gz>.

[Pax97b] V. Paxson, *Automated Packet Trace Analysis of TCP Implementations*, ACM SIGCOMM 1997 Proceedings.

[She90] Scott Shenker, Lixia Zhang, and David D. Clark, *Some Observations on the Dynamics of a Congestion Control Algorithm*, Computer Communication Review, Vol. 20, No. 5, October 1990.

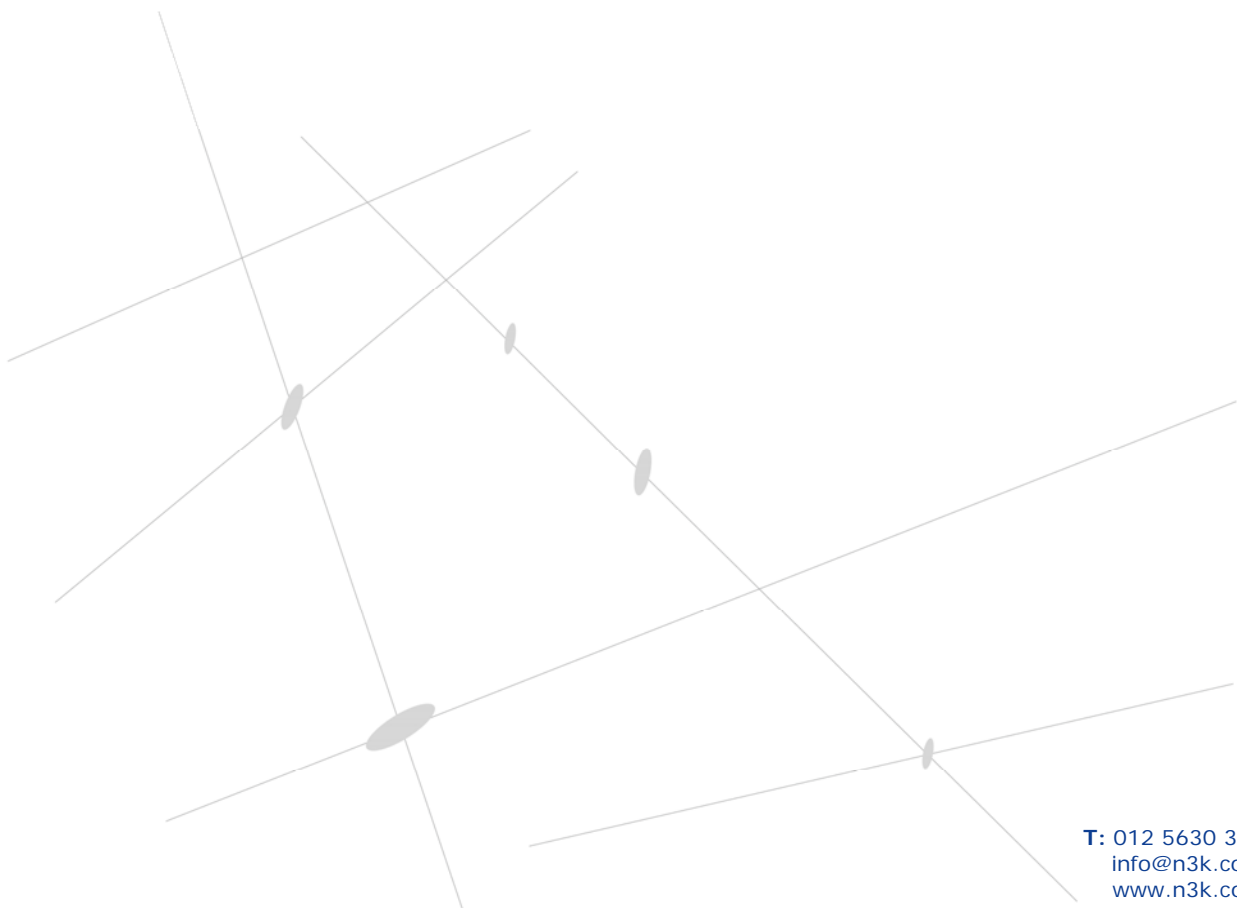


[She91] T.J. Shepard, *TR 494 TCP Packet Trace Analysis*, February 1991, http://ana-www.lcs.mit.edu/anaweb/ps-papers/_TR_494.ps.

[Stev94a] TCP/IP Illustrated, Volume 1, The Protocols, Chapter 20, Section 7, "Bulk Data Throughput", Addison Wesley, 1994.

Copyright © 2000, Lucent Technologies NetworkCare

This is an unpublished work protected under the copyright laws. All rights reserved.



T: 012 5630 3700
info@n3k.co.uk
www.n3k.co.uk